University of Fort Hare
Together in Excellence

Integrating Legacy Applications into Service Oriented Architecture Middleware

A thesis is submitted in fulfillment of the requirements for the degree of

Masters of Science

In

Computer Science

By

Makaziwe Makamba

Supervisor: Prof. Mamello Thinyane

Computer Science Department
Private Bag X1314

ALICE

5700

December 2012

Declarations

I, the undersigned hereby declare that the work that is contained in this thesis is my own original work, and it has not been submitted to any educational institution for similar or any other degree. Any information that is extracted from other sources is acknowledged.

Name: M. Makamba

Date: December 2012

Dedications

I dedicate this work to my mother, Thobeka Makamba, who has been there for me, supporting and guiding me. Thank you Mom for the unwavering love that you show me. To my Dad, Zamikhaya Gxabe who has been there for me, Thank you “Khaya” for every support and motivation that you have given me. I also dedicate this to my late brother, Akhona Makamba, who believed in me, who had amazing support in every direction that I took, who motivated me in all spheres of my life. To my lovely sister Nxonxo and Lazola Makamba, you guys know how to keep me smiling; thank you so much for the support that you gave me. I love you guys dearly. To all my family thank you for being there for me. I also thank everyone who supported me during the course of my study.

Acknowledgements

I would like to thank God Almighty for giving me strength throughout this journey, without you Lord I can do nothing.

There is no achievement in this life without the help of many known and unknown individuals who have contributed to our lives. This work is the result of a lifetime of learning and development from supervisors, friends, colleagues, family and supporters who invested their time and energy in my life, I am grateful for the unwavering support. We are all the sum total of what we have learned from other people, and we owe any measures of success to the input of those people.

Here are the people who made this work possible, and they were there during the inception, incubation and the development of this project.

To my supervisor Prof Mamello Thinyane, thank you for your support, encouragements, guidance, willingness and perseverance on making this work possible. Thank you so much.

To my Mother who supported me throughout this journey, thank you.

To my Dad and family, I'm grateful for the support and encouragements.

To my Dearest friends Bulie and Thembelani, Thank you so much for the support, motivation and for being there for me. I love you guys….!

I would also like to thank my sponsor Telkom, without you my dream of pursuing my Master degree would not have been achieved.

To all my friends and colleagues in Computer Science department thank you so much for your support. Without your help this work wouldn't be complete. To Norbert Jere, Thank you so much for your input in this work. May God bless you!

This work was undertaken within the Telkom Centre of Excellence in ICT for Development which is funded by the DTI through the THRIP programme and Industry Partners.

Abstract

Information and Communication Technology (ICT) is a dynamic approach that is widely recognized as an innovative and powerful tool for socio-economic development, it is a key catalyst for the emergence of knowledge economy. ICT have been used to develop applications, promote transparency and efficiency in multiple services such e-Learning, e-Government, e-Health and e-Judiciary especially for Marginalized Rural Areas (MRAs). The ICT approach is designed to bridge the digital divide. This approach has been widely deployed in many programs and it has led to the development of a new field which is Information and Communication Technology for Development (ICT4D). Within the context of ICT4D there are arrays of e-services that have been deployed to improve the impoverished communities. Some of these applications have failed to bring the changes that were designed to bring in the community due to the use of old architectures. There is therefore a need to develop a system that will integrate legacy applications into contemporary architectures. To solve the problem of the legacy applications we have developed TeleWeaver Service Oriented Architecture (SOA) middleware into which we integrate an e-Commerce and e-Learning applications into SOA middleware. For this integration system there are specific technologies that were used to integrate legacy applications into SOA middleware: RESTful web services using the slim API, SOAP via Nu-SOAP technologies were used to integrate these legacy applications. Specific methodologies were used to achieve the objectives of this research. The literature review, brainstorming, interviews and development of the system are some of the methods that were used to achieve the objectives of this study.

The research methodology is mainly through experiments and to study TeleWeaver SOA middleware architecture. Interviews were conducted to analyze and understand the community needs, since the application discussed in this thesis is tested and implemented for a rural community. The community is called Dwesa, and falls under the ICT project within the Siyakhula Living Lab (SLL). A basic prototyping and Unified Modeling Language (UML) was created to design the system.

This thesis presents the design and implementation of a system that integrates legacy applications into an SOA middleware that brings flexibility and effectiveness to these ICT e-services. The research focuses on integrating legacy applications into Service Oriented

Architecture (SOA) middleware. It seeks to bring flexibility to e-services that are developed for MRAs. The use of SOA architecture that supports re-usability and interoperability of application provides effectiveness to the e-services. The novelty of the system is in its flexibility, usability and sustainability. SOA is an approach that provides a separation between the interface of the service and its underlying implementation. One of the achievements of the integration project is its ability to connect to SOA middleware. This increases the effectiveness of these e-services. The usability and performance evaluations are conducted to test and evaluate the system within the SLL on the TeleWeaver platform.

Publications

The following publications were submitted during the course of this research:

M. Makamba and M. Thinyane. *Implementation of an Adapter Component to Integrate Legacy Applications into an SOA Middleware. Southern Africa Telecommunication Networks and Applications Conference – SATNAC, East London, Eastern Cape, South Africa 2011.*

M. Makamba, N Twele, Miss D Masuku, S Lutshete, L Sonamzi and M Thinyane. *Enhancing Information and Communication Technology Solutions for Rural Communities. IST African Conference, held in Tanzania. 2012*

List of Acronyms

API	Application Programming Interface
	Development
DOSGI	Distributed Open Service Gateway Initiative
GWT	Google Web Toolkit
ICT	Information and Communication Technology
ICT4D	Information and Communication Technology for
My-SQL	My Structured Query Language
OSGI	Open Service Gateway Initiative
OTS	Off the Shelf
PHP	Hypertext Preprocessor
RHS	Reed House System
SLL	Siyakhula Living Laboratory
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSO	Single Sign On
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Service Definition Language
XML-RPC	Extensible Markup Language- Remote Procedure Call

Table of Contents

Table of Figures

Table of Listings

List of Tables

1. Chapter 1: Introduction

This introductory chapter presents both the aim and the context of this research. In its discussions, it provides a detailed account of the research problems together with the motivation leading towards engaging in the research. While discussing these aspects, the research brings up the general overview of fundamental reasons for integrating legacy applications into an SOA middleware. The key objectives of this research are also provided with a detailed methodology. The chapter also outlines the general research overview as well as its research deliverables.

1.1 Introduction

Application modernization is a main arena that is used by many companies to reduce the cost of ownership; enhancing application flexibility, as well as the performance, efficiency and usability of legacy applications. It is a broad term that covers the design of the applications, the analysis of the architecture, implementation and the methodology of modernizing these legacy applications (HP services, 2009). It includes the transforming of application, data and infrastructure from a rigid legacy environment to a modern environment taking advantages of today's open standards applications which includes Service Oriented Architecture (SOA) (HP services, 2009).

This arena is used to improve the structure of existing applications, to enhance the performance of legacy applications and to improve the value of the existing systems. Application modernization strategy implies both acquisition and deployment of modern technologies along with their associated skills to replace legacy applications (Howard, 2009).

Legacy applications have become a serious problem in many organizations. They present a series of challenges to the organizations. They are difficult to modify, require a high cost of maintenance and they do not meet today's business demand.

For these stated reasons many organizations are considering a move to new architectures that offer benefits such as reducing the cost of maintenance, increasing agility and improving respectively the compliance. However it is important to modernize applications based on an architecture that is built on open standards. Developing new applications uses current architectures is expensive and time consuming; hence modernizing the existing applications is a cost-effective solution which provides a better way to upgrade the standard of the legacy applications which is cost effective. Instead of replacing the old system, modernizing legacy application seems perfect as it revives the core application systems by integrating them into new platforms, and gives them modern interfaces. To this regard maintaining a stand-alone legacy application is not only time consuming but also very expensive, hence application modernization is preferable in many companies.

The application modernization approach has been considered in the Siyakhula Living Laboratory (SLL) intervention, to provide sustainability and flexibility to the existing applications. This approach is highly adopted in this intervention as it enhances the performance of legacy applications. The next chapter gives explicit details of the application modernization approach under SLL context. The following section provides details about the research context.

1.2 Research Context

This research takes into context the operations of the Siyakhula Living Lab (SLL), which is based in Dwesa, a marginalized rural community that is located on the wild coast of the former Transkei in the Eastern Cape Province of South Africa. The Siyakhula Living Lab (SLL) is an Information and Communication Technology for Development (ICT4D) intervention that explores the use of Information and Communication Technologies (ICTs) in marginalized rural areas through the development of different applications to improve the impoverish communities.

The major focus of the SLL is intended to develop and field-test the prototype of simple, cost effective and robust integrated telecommunication platforms to be deployed in marginalized rural communities (Jere et al., 2009b; Njenje, 2008; Tarwireyi et al., 2007). Its objective are to build a large capacity of technically skilled human resource in the field of ICT4D and e-Commerce platform and develop the community through the exposure of multi-function distributed communication platforms such as e-Commerce, e-Learning, e-Government and e-

Health- which are the applications that allow easy access to the information and to the community.

One of the aims of the SLL is to provide new technologies and skills to the Mbashe Community in Dwesa with the hope of developing and improving their lives and reducing the digital divide between the communities and the technologically advanced professionals (Pade el at., 2009). One of the predominant needs of the community is to be able to access information without any difficulties. The details on the activities of Dwesa and the inter-actions as well as intra-actions inside the SLL will be discussed in the second chapter of this research report.

The research projects that have been conducted discovered that maintaining existing applications is expensive and time consuming; hence many companies have decided to modernize by integrating them into the new (SOA) platforms such as middleware and create web services, to improve the existing application structures and their functionality.

The main focus of this research is on adopting a platform that allows legacy applications to be re-usable, valuable, cost-effective and flexible while expanding their functionality, through modernizing and integrating these applications. The application modernization concept is considered as a better option than the total replacement of the existing applications.

Application modernization is an interesting major phenomenon in ICT4D interventions, enterprises and in modernization environments because of its offers or advantages in renewing or transforming applications. Application modernizations have enabled the sustainability of ICTs in rural areas and have increased the number of applications that have been developed for rural areas. There are many stand-alone applications that are developed within the Siyakhula Living Lab, which need to be integrated into Service Oriented Architecture (SOA) middleware to reduce the cost of maintenance and improve the efficiency of these applications. This research focuses on integrating the legacy applications into the SOA middleware platforms.

The SOA approach is preferable when modernizing legacy applications. The SOA has become vital architecture that provides a flexible access across the various applications in a distributed system. It also supports the different integration platforms as it support communication across the different platforms and services. Therefore, organizations use it for the modernization of their existing applications.

This approach is adopted in ICT4D to integrate legacy applications into a middleware platform. The SOA platform allows ICT4D e-services to be deployed in a flexible and re-usable manner. SOA is also an architecture that allows a great flexibility in adapting the Information Technology (IT) infrastructure to support business needs. It is an architecture whose goal is to achieve loose coupling among interacting software agents (He Hoa, 2003).

The SOA is highlighted as a very user-friendly mechanism to modernize the legacy applications and to support various communication platforms of the services (He Hoa, 2003). It allows e-Services to be deployed in a distributable and appropriate manner. In view of this research, the SOA approach is used to modernize the existing applications and integrate legacy applications into the new middleware platforms, which allow the interoperability and re-usability of the existing applications. All this is done in order to leverage the existing services and to improve the performance of the legacy applications, along with minimizing the problems that are presented by legacy applications.

1.3 Research Problem

The main problem that this research aims to solve is how to integrate legacy applications, specifically Os-Commerce and e-Learning applications and third-party applications into SOA middleware.

The above stated legacy applications are the stand-alone applications that have been deployed under the Siyakhula Living Lab (SLL) intervention. The major problem with these applications is that they are unsustainable, inflexible and are not effective. Since they require high maintenance cost. They are difficult to modify and to meet ongoing business demands. They do not adequately meet today’s compliance demands.

Ron Wertlen (2010) stipulates that the effectiveness of these legacy applications can be improved only if these applications can collaborate with each other in the same environment. Hence he suggests the development of a TeleWeaver middleware platform for the integration of these. Since there is a problem with disparate systems when it comes to maintaining and modernizing them, it is difficult to improve the stand-alone application performance as it requires the understanding of each application's architecture and language of each application. For the community to access information effectively over an e-Service platform, there is a need to modernize these existing applications.

There is therefore a need to provide mechanisms to integrate legacy applications into these new architectures in a manner that leverages the services capabilities that are available in the new architectures. At the same time there is a need for improving the e-Services that are offered on these new architectures/platforms. However, TeleWeaver SOA middleware have been deployed in Reed House System (RHS), which is a company that is affiliated with SLL, to create a platform that will allow these applications to collaborate with each other.

1.4 Research Questions

This research aims to answer the following questions:

1. What are the advantages and benefits of integrating legacy applications into SOA middleware?

2. What is the best way of integrating the 3rd party applications specifically e-Learning, Os-Commerce application and OTS packages into SOA middleware?

3. What are the different techniques focusing on Nu-Soap, XML-RPC, Soap, Slim and Rest that can smoothly integrate legacy services into SOA middleware?

1.5 Aims and Objectives of the Project

The purpose of this research is to investigate the best ways of integrating legacy e-Services into SOA middleware within the SLL context. The SLL has many software applications that are stand-alone and there is a middleware platform that is developed for the integration of legacy applications. The problem is how to integrate these existing applications into the new single platform. The focus is on identifying the application modernization techniques that will allow the smooth integration and reduce the risk of losing old applications' context along with how to integrate the legacy applications into a TeleWeaver SOA middleware.

The other purpose is to undertake the integration of related third-party applications into SOA middleware. Therefore there is a need to evaluate and test the effectiveness of this integration process. To implement and deploy this system will improve the performance of the existing applications, while increasing the scalability and re-usability of these legacy applications. To implement a system that allows existing e-Services to be interoperable, flexible and easy to use to the end users will increase the effectiveness of these legacy services.

The main objective of this research is to integrate legacy applications specifically Moodle and e-Commerce applications into TeleWeaver SOA middleware.

The following are the sub-objectives of this research:

1 To identify the advantages of integrating legacy applications into SOA middleware.

2 To investigate integration and application modernization techniques.

3 To identify the best way of integrating e-Services specifically e-Commerce and Moodle applications into TeleWeaver SOA middleware.

4. To investigate web services technologies, Application Programming Interface (API), library that can be used to integrate the e-Services into SOA middleware.

5. To implement the integration of e-Commerce and Moodle applications.

6. To evaluate the effectiveness of the integration system.

1.6 Research Methodology

The methodology used in this research includes three main areas.

- Comprehensive literature reviews, based on understanding the concept of application modernization, integration platforms and the web services.
- Conduct different experiments, to study and understand the architecture of the TeleWeaver SOA middleware.

➢ Analyze the environment of the new platform and system requirements.

The second area of this methodology is to engage in interviews to develop the application modernization and integration of e-Services into new SOA platform. The interviews are conducted to analyze and understand the needs of the community, since these applications are developed for the Dwesa community, it is essential to know what the community wants.

The interview trips are conducted in order to understand the community and to analyze and understand the architecture and the way that TeleWeaver Middleware works. A Unified Modeling Language (UML) will be used to design the system. Usability evaluation and performance evaluation are performed to test and evaluate the system. The system will be tested to validate it against the specified system requirements. Functional testing will be carried out to investigate the functionality of the system as well as usability testing that is conducted to check if the system is user-friendly. This is also used to check the performance of the integrated applications. The last step is to implement the system and evaluate the effectiveness of this system.

1.7 Overview of research objectives

The following table will give an overview of the research objectives, how to address them and the chapter where these objectives are addressed.

Table 1 Research Objectives overview.

Research objectives	How to address	Chapter addressed
1. Identify the advantages of integrating legacy applications into SOA middleware.	This will be achieved through literature review	Chapter Two
2. To investigate integration and application modernization techniques.	Through literature reviews and experimentation	Chapter Two
3.To identify the best way of integrating e-Services specifically e-Commerce	This will be achieved through development of	Chapter Five

and Moodle application into TeleWeaver SOA middleware	the system	
4. To investigate web service techniques, API, Libraries that can be used to integrated legacy applications into SOA middleware	Literature Review and development of the system	Chapter Two and Chapter Five
5. To implement the integration of e-Commerce and Moodle application.	Through developing a working system	Chapter Five
6. To evaluate the effectiveness of the integration system.	Through system testing, experiment and evaluation.	Chapter Six

1.8 Research Motivation

This research is motivated by different factors including the extensibility of TeleWeaver functionality to provide the best and flexible integration techniques of legacy applications (e-Learning and e-Commerce) into SOA middleware. Application modernization is also another factor that motivated the development of this project. Improving the functionality and the architecture of the legacy applications is another factor that motivated this project. The benefit of using modern SOA is also another factor that motivated this project.

Transforming the traditional applications into new architectural platforms and the availability of TeleWeaver SOA middleware that allows different applications to be integrated into a single platform and to be reusable is also one major motivating factor. Another inspiration to this research is the leveraging of legacy applications to make them flexible, interoperable and extensible. The need is to improve the visibility of these legacy e-Services and hence to improve the usability of e-Commerce and e-Learning applications. Providing a user friendly system that provides a great satisfaction to end users or the community also motivated the development of this research. Providing the Dwesa community with a quick and user friendly system is another factor that motivated this research. An e-Commerce shopping portal has been developed for Dwesa community, but the utilization has been minimal due to low usability of the art and crafters interface on the portal. Therefore modernizing and integrating this application into a new SOA platform will be effective in improving the functionality and accessibility of this service. Since e-Learning is an application that equips the community with knowledge; there is a need to provide the mechanism to integrate this application so that the community can easily access the information from this service.

1.9 Dissertation Overview:

The structure of the research is structured as follows:

Chapter 2 is a literature review. It reviews literature that is relevant to the application modernization and the integration of legacy application into SOA middleware. It also reviews the ICT4D concept and describes the number of technologies used in ICT4D context. Further, it discusses the new architecture platform that is used in SLL.

Chapter 3 provides a detailed discussion of the methodologies that are used in integrating the e-Service legacy into TeleWeaver middleware. It also explains the new SOA middleware platform that is used to integrate these applications, and the technique used to modernize legacy applications.

Chapter 4 deals with a system design and provide the specification keys of the system requirements. The chapter illustrates the design of the system. It provides the functional and nonfunctional requirements of the system. It explains the whole system through the use of UML and use case diagram.

Chapter 5 covers the implementation of the system, describing the implementation of the system and explaining the codes and PHP scripts that allow the smooth integration of legacy application.

Chapter 6 includes system testing and experimental results. In this chapter the research evaluates the effectiveness of the application integration. It provides the feedback that is received from the end-users.

Chapter 7 provides the discussion and conclusion of this project. It gives an overview of the achievements and the summary of the research. This chapter also discusses the best ways to improve the integration of legacy application. The summary of all challenges and the successes of this system are discussed in this chapter. Lastly it gives the summary of the entire research project.

1.10 Conclusion

The problem facing the SLL e-Services has been identified; hence this research aims to integrate and modernize these legacy applications into TeleWeaver SOA middleware. The integration of ICT services is intended to make the existing application to be flexible and interoperable. The application should improve the performance of the legacy applications while providing the end-user with a user-friendly system and a modern user interface. The next chapter discusses the literature review that was conducted during the research process

2. Chapter 2: Literature review and Technology review

This chapter reviews the literature that is relevant to the integration of e-Service applications into SOA middleware platform. It also discusses more about application modernization, how application modernization has been done previously and how it is currently done. We looked into techniques that are used to modernize legacy application and the tools that can be used to integrate legacy applications and OTS packages under SLL intervention. We also looked at the benefits of modernizing and integrating legacy applications. We have provided explicit details of SOA middleware platform that is used within the SLL intervention and how the application integration is done.

2.1 Introduction

This chapter introduces the area where this project is conducted and applied, which is Dwesa a Marginalized Rural Area (MRA) in the Eastern Cape region. It elucidates more about application modernization techniques that can be used to modernize the existing e-Services that have been developed for SLL, the advantages and benefits of application modernization on the SLL intervention will also be discussed. It also gives a brief explanation about application integration, Service Oriented Architecture (SOA) platforms, e-Services, ICT4D platforms, as well as the projects that have been developed for Dwesa community.

An overview of our test bed SLL is given in this chapter, which is a lab where a number of e-Services have been developed. It goes into detail on the role of application modernization in ICT4D in MRAs and how application modernization and integration application under ICT4D assist to overcome the challenges that MRAs encounter.

This chapter gives the challenges facing the Dwesa community and protocols for eliminating those challenges and ways of minimizing the digital divide through the use of ICT. In addition to that, it discusses the challenges of using a traditional architecture in Dwesa, and also gives a brief discussion of TeleWeaver, which is a middleware that has been developed for SLL and the benefits of using TeleWeaver middleware. Finally it will explain more on the benefits of integrating e-Services into TeleWeaver SOA middleware. This chapter discusses more about the application integration and the benefits of integrating legacy applications into SOA middleware and the ways of integrating these legacy applications. The ways which application integration is done in the enterprise environment and how SLL intervention is integrating these applications is also to be explained in this chapter.

2.2 Application modernization

Application modernization is a process of adapting the existing applications to modern standards to improve effectiveness and to enhance the performance of the existing applications (Howard, 2009). Application modernization moves existing application from the legacy hardware to new and innovative environment. It replaces the Information Technology (IT) systems with advanced packages. With the current economic downturn, many organizations are now considering legacy application modernization as an alternative to expensive and risky new-application development (Aberdeen Group, 2009).

Many project such as SLL seek to enhance the effectiveness of legacy applications and to improve, and maintain legacy services for MRAs communities. However, maintaining legacy application and infrastructure software in a legacy environment consumes a disproportionate percentage of IT budget. It is discovered that the average companies, spend from 60 to 85 percent of their IT budget maintaining legacy applications that fail to meet the changing competitive needs of the business (Howard, 2009).

Application modernization is the continuous evolution of an organization's existing applications and infrastructure software, with the goal of aligning IT with shifting business strategies (Howard, 2009). However, this implies that the acquisition and deployment of modern architectures, contemporary technologies along with their associated skill and sets capabilities improve legacy services.

Legacy applications have become a significant problem in many organizations. They are difficult to modify and do not adequately meet today's compliance demands (Aberdeen Group, 2009). For these reasons, many organizations including SLL are considering the move to new technologies and architectures. However, while it is possible for many organizations and SLL intervention to develop applications from scratch, that fully utilize new architecture and new technologies, but the approach is expensive and risky. The flexible strategy that a growing number of organizations are embracing is to modernize their legacy applications.

Application modernization based on an open architecture offers the benefits of reduced total cost of ownership, increased agility, reduced reliance on legacy skill sets, and improved system functionality. It also improves the effectiveness of legacy services and enhances the sustainability of these existing services. It also reduces the cost of maintaining these applications by optimizing business processes to save labor costs, eliminating ongoing maintenance, support fees for expensive, proprietary legacy infrastructure and automating previously manual processes. This is to further reduce the cost of labor and reducing the need to extend or modify applications, through the use of re-usable service-oriented architecture (SOA) components (Howard, 2009).

The cost reduction can happen only when project or labs such as SLL can adhere to key principles; such as leveraging packaged applications, use lower-cost software platforms and consolidating technologies.

To reduce the cost, the use of SOA middleware platform that uses the modern technology such as TeleWeaver Middleware is essential. The modern platforms combined with SOA create the next-generation IT environment or enabler, where orchestrated application components combined with computing resources in multiple locations. This forms a virtual environment with a single point of management, control, and access. The SLL intervention supports exactly such architecture where it can be used by multiple end users.

Many organizations consider using packaged applications, these packaged applications should adhere to SOA standards and work with the organization's underlying architecture.

"Using an assortment of highly proven IT modernization approaches, organizations can consolidate technologies and technology providers while determining the best combination of modernization approaches for each application" (Howard, 2009). The modernization approaches depend mostly on the needs of each project, as each project differs in architecture and programming language.

The Aberdeen group reveals that many companies are seeing substantial reductions in development cost and generating improvement in operational and development agility by modernizing their legacy services and adopting new development processes (Aberdeen Group, 2009).

The Gartner group reported that many companies are leveraging the tools based approach coupled with a strong base of organizational capability to drive down modernization costs and realize substantial improvements in software agility (Gartner Group, 2009).

Application modernization allows applications to be flexible in a manner that will allow developers to modify and integrate legacy applications into an SOA middleware platform (Howard, 2009). Application modernization helps to cut the cost of maintaining old systems and the development cost. Hence SLL intervention is intending to use application modernization techniques to improve the performance of legacy application and to enable the flexibility for the existing applications. Application modernization allows organizations to maximize the use of their existing application assets as they move toward more agile and cost-effective technology environments (Oracle, 2008).

As application modernization allows existing applications to be easily integrated into these new architectures, it also allows the Off the Shelf (OTS) packages to be integrated into the middleware in a smooth and flexible manner. It enables organization to rapidly evolve to business process level as it allows application to be cost-effective. The application modernization projects involve creating new business value from existing applications, incrementally transforming legacy systems into new reusable business components, or leveraging existing enterprise skills and improving productivity (Software AG, 2008). It improves and enhances productivity by leveraging existing skills. Application modernization also offers secure, high-performing services for new customers and enables seamless cross-channel operations.

There are lots of legacy e-Services that have been developed under the SLL context that need to be modernized and integrated into a new SOA middleware platform. In this chapter we discussed the legacy e-Services and their functionality that were developed under the SLL banner, these e-Services are stand-alone and there is a need to modernize in a standardized manner and integrate them into SOA middleware to improve their performances. It is hoped that modernizing and integrating legacy applications into SOA middleware will provide effectiveness and flexibility to these legacy systems. Modernizing legacy application is critical. Therefore the road-map is essential. To design a strategic modernization road map, one have to start with an assessment of the current state of the environment and a desired future state vision of the system. In addition, the architectural design must be checked. It is vital for the assessment to consider the following factors as input on a road map (Summa, 2006). Direct and peripheral project goals, functionality and timeliness. It is good to consider internal and external integration points, dependencies between applications and infrastructure components. A well-structured interview, questionnaires, brainstorming and meetings are the best tools used to obtain stakeholder input for road-map decisions and priorities (Summa, 2006). The following section will give details about research area and the e-Services that have been developed.

2.3 Research Location- Dwesa and SLL intervention

Dwesa is a marginalized rural area in Eastern Cape region, it is where this project is conducted and implemented. It is located on the Wild coast of the former Transkei in Eastern Cape of South Africa. Dwesa is an area where lots of people are illiterate but creative; they have a skill of making art craft (Dalvit, et al. 2007). It is a representative of many marginalized rural realities in South Africa. Dwesa has a magnificent nature reserve where lots of tourists comes and visit. With such areas having an illiterate community and a nature reserve on it, the University of Fort Hare and Rhodes University have come together and developed as many e-Services as possible, such as e-Commerce, e-Health, e-Government and many more that will help this community to be able to sell their products online.

Dwesa is characterized by a lack of infrastructure, poverty, lack of services and isolation in terms of knowledge and information (Human resource council, 2005). The lack of basic access deprives the community from knowing what is happening globally, or around the world. The lack of access to information and knowledge is a major factor for the digital divide.

2.3.1 Siyakhula Living Lab

Siyakhula Living Lab is one of the Southern African labs. It is based in the Eastern Cape, of South Africa. It is a multi-stakeholder operation that consists of academia, industry, government and marginalized communities to assist user-driven innovation in the Information and Communication Technology for Development (ICTD) domain, and it is supported by COFISA and RSA Department of Science and Technology (Dalvit et al., 2007). Siyakhula began in 2006 as a joint initiative between the Telkom Centres of Excellence at Rhodes University and University of Fort Hare, with an aim of developing and field-testing robust, cost effective and integrated e-business/telecommunication platforms (Sustain Living lab, 2011). University of Fort Hare and Rhodes University are joined together to run the projects under the banner of SLL, means "we are growing together", as a part of ICT for development in the Eastern Cape region under the Dwesa Community.

The main idea behind the SLL is to facilitate activities in rural ICT applications and services in a multi-channel environment and deep rural context. The main objective of SLL is to develop and field test the prototype of a simple cost-effective and robust and integrate e-services and telecommunication platforms to be deployed in a marginalized and semi marginalized rural communities (Living Labs networks, 2011). Another objective of SLL is to enlarge capacity of technically skilled human resources in the field of e-services and ICT4D, also to evolve to a multi-functional distributed communication platforms and centralized models. The SLL intervention aims to deliver real capability and develop real skills while demonstrating the practical validity of the use of ICT4D in government. It provided the community with a distributed access to various services such as e-Commerce, e-Government, e-Learning and e-Health. SLL is located under the Dwesa community. SLL has made a tremendous progress in improving the quality of life in Dwesa community through deployment of ICT infrastructures, such as networking, e-services, e-skills and Digital Access Nodes.

2.3.2 ICT for rural areas

Information and Communication Technology for Development is focusing on supporting strategies for building a digital economy comprising of: e-finance, e-transactions, e-

Commerce, e-trade and e-content and creating an enabling environment for the knowledge economy (World Developments, 2009). There is a need to build e-security capacity for securing networks and infrastructure for the knowledge economy, improving government services to citizens and business, and strengthening the role of small and medium-scale enterprises and other stakeholder groups in the digital/knowledge economy through ICTs (World Developments, 2009). In developing economies innovative use of ICT services is changing people’s lives and providing new opportunities. For an example, banking services and job search text messaging services can be delivered through mobile phones and portable devices. Some ICT identified by many authors are: e-Government, e-Commerce, e-Health, e-Learning, e-Agriculture, e-Postal, voice telephone they provide a basic access to information and interactive communication (Dymond et al, 2003).

2.3.3 The Legacy e-Services

There are lots of legacy e-Services that have been deployed at Dwesa which are discussed in the following paragraphs. These e-Services have been developed to help people who live in marginalized rural areas such as Dwesa, who had to travel long distances to obtain required products, and also to help people who have skills of creating art and craft to be self-employed. The explained legacy e-services need to be modernized and integrated into the SOA middleware platform to accelerate their performances and functionalities.

2.3.3.1 The e-Commerce Services

The e-Commerce application is one of the applications that have been implemented for the Dwesa community by one of the Fort Hare and Rhodes researchers. The e-Commerce services involves the purchase of goods and services over computer networks by businesses, individuals, governments or other organizations (Schneider, 2003). The aim of this application is to generate income in these communities. Since art and craft makers in Dwesa with skills of making art craft, it was seen as a need to develop a service that allows people to sell everything they make online. This application helps the Dwesa community to sell their artifacts, by doing that it makes them self-employed and create lot of attraction from tourist, as well as creating a job opportunities for community. This elaborates how ICT4D and SLL intervention has helped rural areas create job opportunities and minimize crime because youth is occupied by designing art and craft and earning a living. Some of the people in

Dwesa are old but can write poems and sell them online. The propagation of e-commerce applications in rural areas has resulted to the increase among a number of community members who sells the products. This exposes the advantages of using ICT in communities.

2.3.3.2 The e-Health Service

The e-Health is one of the services that have been implemented for this community this application helps people from this area to be able to buy medicine online without travelling 100km to go to town and buy medicine. The ICT is improving these communities by developing such services. Government and researches are also trying to bridge the digital divide, by developing applications that will make life easy for people who are living in marginalized rural areas.

2.3.3.3 The e-Government Service

This application provides services to the user to be able to access the government services online. Since it is a long distance to go to town when you are at Dwesa, this kind of application will help people to make the child birth certificate online, it will not be necessary to travel 100km to make a child birth certificate.

2.3.3.4 The e-Judiciary Services

The judiciary system is the entirety of courts and judicial authorities in a state or in another sovereign organisation such as the European Union (E-European, 2006). This system is for handling legacy issues and to resolve legal disputes and to ensure that the law is applied correctly and coherently. Dwesa legal issues have always been discussed at the chief's house under a tree, and once the issues have been resolved, the people are dismissed (Scott, 2010). However after a short period of time nobody remembers everything correctly, that results in a contradiction between the community members because each has his or her own way of theory of what was discussed the last time. The e-Judiciary services augments traditional judiciary processes by making available a way of safe guarding vital legal data, and guaranteed persistence and availability of data. It also makes legal information easily available to the public (Scott, 2010).

All these above services were developed to improve the community and to bridge the digital divide. These applications have been developed to enable rural community to access information of different sectors over the network, however the problem with these e-services are disparate and that makes it difficult for them to be modified and maintained, hence this project is focusing on the integration of these legacy applications into the SOA platform. However there are many technologies and methods that need to be implemented to integrate and modernize the existing systems which will be discussed.

2.3.4 SOA under SLL intervention

There is a lot that has been done by SLL to modernize the existing applications, such that there is new SOA middleware that has been developed to integrate these applications under SLL banner. SOA has provided a flexible mechanism to integrate these existing services. SLL has been using new architecture such as SOA to modernize the legacy applications under ICT4D context. The evolution of the architectures presents the mechanism that makes legacy applications re-usable, inter-operable and valuable, while extending the functionalities of these services. A lot has been done by SLL intervention and ICT4D to improve the life of MRAs inhabitants, starting from developing e-Services to developing middleware platform for these areas. This shows the importance of embracing innovative architectures and technology for MRA environments.

A growth propagation of ICT4D interventions has necessitated the exploration of innovative solutions for the provision of e-Services in marginalized rural areas. However, ICT itself cannot be elixir to all problem that are faced by marginalized rural areas, it provides new communication channels for rural areas and offer new approach to reduce the gap between ICT illiterate and ICT literate communities; but communities have to be involved by cooperating with the SLL to promote changes, hence Fort Hare and Rhodes have developed e-Services platforms to enhance the standard of living in those impoverished communities.

2.4 Technologies for Modernizing and Integrating Legacy Applications

There are several methods and technologies of modernizing and integrating legacy applications into an SOA middleware. One of the tools is SOA architecture. SOA is an architecture that allows services to be integrated, re-usable and interoperable (Bradley, 2008). SOA is a vital tool that will be used to modernize these applications as it will allow them to have a great flexibility (Bradley, 2008). Web services are also phenomenal protocols that are essential for modernizing and integrating existing services. Different methods have been implemented in the enterprise environment to address the problem of application integration; hence this project is exploring ways to integrate legacy applications into SOA middleware platform under SLL intervention. The following section will describe the tools and the methodologies of modernizing and integrating legacy applications into an SOA middleware.

2.4.1 Service Oriented Architecture (SOA) Platforms

Service Oriented Architecture (SOA) is becoming a great innovative architecture in software industry, which supports re-usability and interoperability of applications (Makamba and Thinyane, 2011). It is an attractive architecture as it lowers the integration costs, for greater innovation and transformation at the line of business level. “SOA is an approach for designing, implementing, and deploying information systems, such that the system is created from components, implementing discrete business functions” (Sonic Software and AmberPoint, 2005). It is built on the standards of the World Wide Web leading to cost-effective implementations on a global basis with broad support by vendors (Sonic Software and AmberPoint, 2005). However, SOA have services that are loosely coupled components that allow more flexibility than older technologies with respect to re-using and re-combining the services to create new business functions, both within and across enterprises. It creates designs which embody business processes and enhance the ability to outsource and extend processes to business partners. It enhances legacy applications and processes so that the usefulness of existing services can be preserved and improved.

SOA architecture is more concerned about enhancing business performance by re-using the legacy applications and by allowing the easy integration of legacy services, along with several dimensions including cost reduction and streamlining the implementation of new business models. “SOA is an evolution of distributed computing designed to allow the interaction of software components, called services across the network” (Sonic Software and AmberPoint, 2005). The SOA components, called services, can be distributed across geography, across enterprises, and can be reconfigured into new business processes as needed (Sonic Software and AmberPoint, 2005). These services are message oriented and have interfaces, can be on heterogeneous systems across networks and geography providing platform independence and location transparency.

These services communicate by standard protocols providing a broad range of interoperability and are commonly called web services. The web services will be discussed later in this chapter. The contextualization of an SOA platform is revolutionizing how distributed applications are organized, also to offer opportunities on modernizing legacy applications. The sustainability of legacy application projects in SLL is a critical factor; however the re-use of software components should be seen as a driver towards both social and financial sustainability. The SOA architecture allows legacy application systems to be integrated as a services that leverage’s existing investments. It support re-usability and extensibility of services, it helps expand and enhances business models. It helps applications to be cost-effective, flexible and sustainable.

The SOA platform provides mechanisms to front-end legacy applications behind a services interface with no changes to the existing system. SOA applications components are usually designed to be event-driven, they respond to the messages as they arrive. The implementation of SOA components relies on skills, methods, and an SOA infrastructure, for them to support the SOA applications perform the work in a reliable, flexible, scalable, and secure manner.

It is essential for these e-Services that have been developed for SLL intervention to embrace this contemporary architecture to upgrade the standard of these e-services. Since these e-Services that have been developed for the Dwesa community are using the traditional architecture, that makes them to difficulty to maintain and sustain, therefore using SOA platform will improve the existing systems. One of main goals of SOA is to allow this business agility in different ways and advanced technologies; different due to the capitalization on Web standards, inherent flexibility of the SOA design, due to the inclusion of legacy systems, and to the availability of off-the-shelf SOA infrastructure and services (Software, AG. 2008)

The use of SOA can accelerate the growth and proliferation of e-Services for Marginalized Rural Areas (MRA's) by enabling easy integration of third-party services through Web Services interfaces. The introduction of SOA provides explicit details of the technology that can be used to meet the goals of SLL intervention.

2.5 Origin of SOA

The former analyst at Gartner, 1994 Alexander Pasik, invented the term SOA for a class on middleware that he was teaching in 1994. Alexander was working on SOA before the extensive Mark-up Language or Web Services were invented, but the basic SOA principles have not changed. Pasik was driven to create the term SOA because “client/server” had lost its classical meaning (Josuttis, 2007). Many business companies have adopted the SOA platform because of its offerings.

2.6 Reason for Using SOA approach

The reason many industries use SOA is because it promotes the notion of modularity, providing overwhelming flexibility and superior economics for addressing business demands and improving applications. SOA provides a conducive environment where developers offer their product as re-usable and interoperable services (Josuttis, 2007). It is a strategic approach to the future of business, and also allows customers to consume services such as web based application and be alert of all available services. The Figure 2.1 below illustrates the way SOA functions and the reason organizations choose SOA over old architecture such as point to point platform.

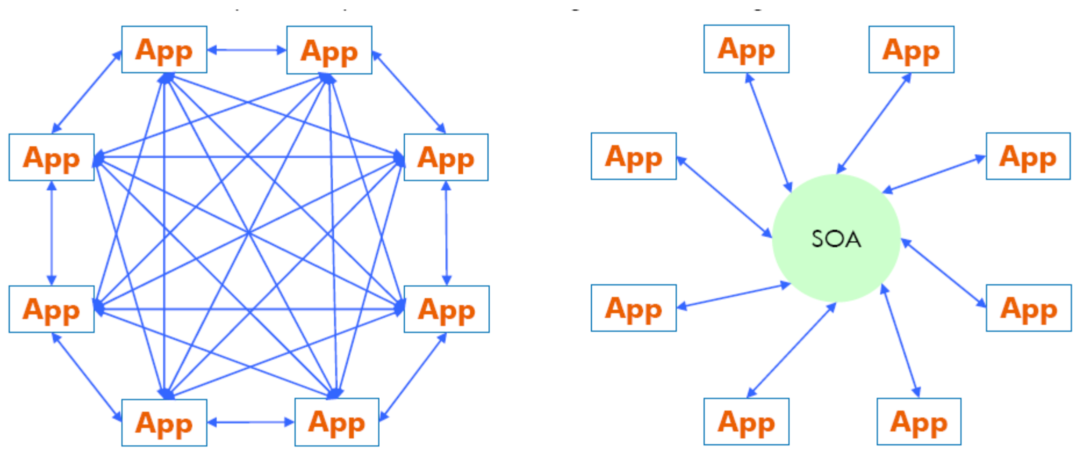


Figure 2.1 An SOA and point to point view integration (Kumar, et al. 2006)

Figure 2.1 shows the benefits of using SOA for integration. This highlights the advantages of SOA: it enables the easy process of integrating widely disparate system applications, the SOA allows loose coupling of services. It enables application functions to be distinct units, which makes for the developer to re-use the application (Kumar, et al. 2006). SOA establishes visibility into the relationships and interdependencies of components so that developers can foresee the impact of changes within the application, in that way it accelerates the developer's productivity (Kumar, et al. 2006). This visibility is important in reducing the risk associated with change and ensuring the desired results. The next diagram depicts the way that old architecture was working. This is an example of many e-Services that are developed for SLL intervention. These services have similar functionality that needs data redundancy, such as login.

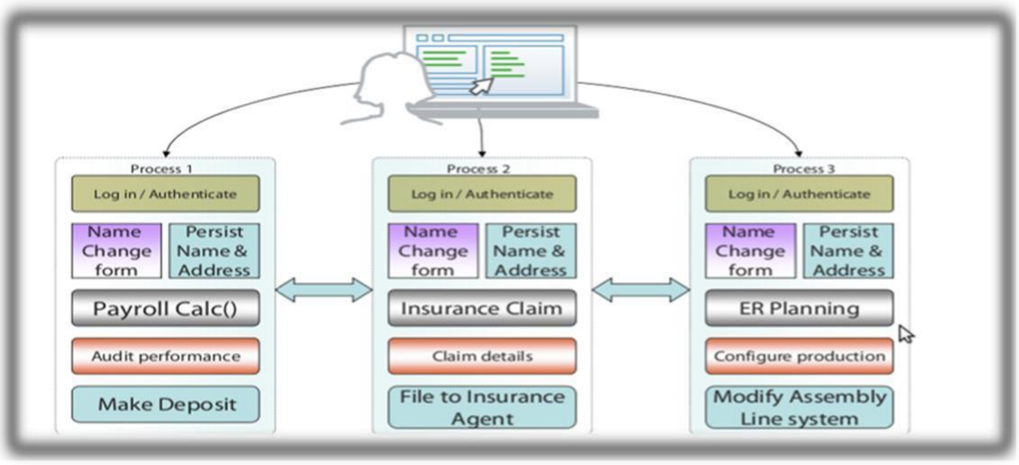


Figure 2.2 Old system without SOA platform (Reitman, et al. 2007)

This diagram show how the repeated data happens when using old architectures. The users have to enter credentials for every service, which is data redundancy and it consume the time of the client. The disadvantage of using these old architectures in MRAs is because many people in rural areas are old, they cannot remember password for service. Hence we moved to the new architecture such as SOA.

Figure 2.3 below illustrates the advantages of SOA platforms. This diagram gives more details about the SOA platforms. It also shows why SOA is a preferable architecture in many organizations.

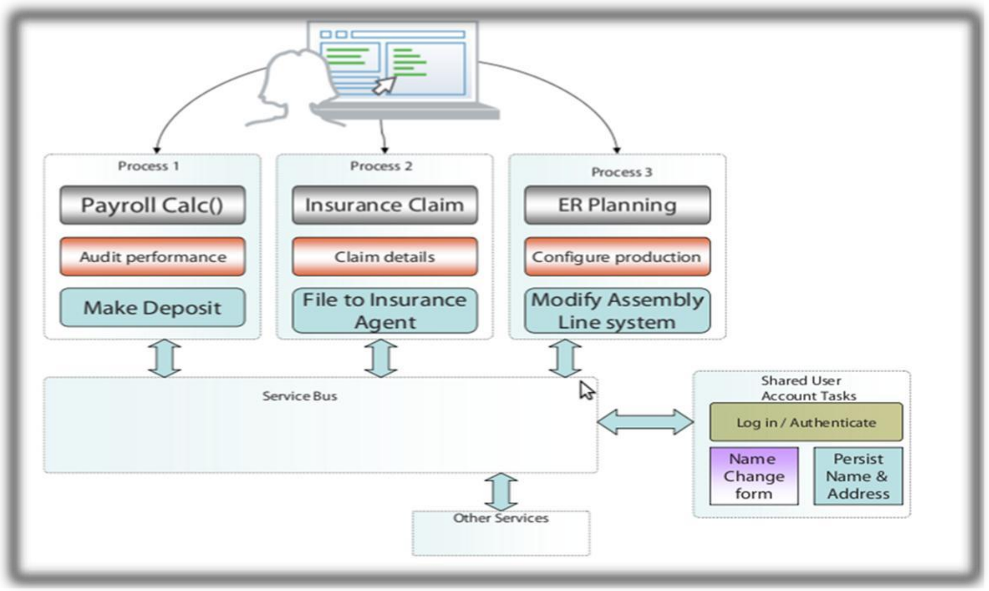


Figure 2.3 System with SOA Contemporary Architecture Adopted (Reitman, et al. 2007)

Figure 2.3 demonstrate the main reason that motivates many organizations to use current architectures such as SOA. It is because of their ability to allow the Single Sign On (SSO). This is an architecture that allows the user to enter credentials for only one service, and then gets into other services automatically. This is the main reason why we use SOA middleware in SLL intervention; this eliminates the data redundancy and saves time for the user to search for the intended information.

The SOA platform also focuses on allowing users to seamlessly integrate fairly large existing e-Services functionalities to form *ad hoc* applications, which are effective and sustainable. It also allows applications to be modernized with infrastructure and it supports different integration platforms. One of the focuses of SOA is helping developers on creating shared services that are necessary for sharing within a reasonable domain.

Service Oriented Architecture is an architectural paradigm that may be used to build infrastructures enabling those with needs and those with capabilities to interact via services across disparate domains of technology and ownership (Reitman, L. 2007). The other benefits of using SOA are risk reduction – control dependencies; manage impact of change; enforce policies, manage SLAs (Oracle and IDG, 2009). Business value ensures project investments yield business value. Cost effective and efficiency, promote consolidation, standardization and re-use.

Re-use is a core principle of SOA, but users need to be able to discover existing services that might already meet their requirements (Oracle and IDG, 2009). ICTD has selected SOA as it is an essential platform to achieve measurable, sustainable, effective, lowering cost and improving efficiency and increase visibility of the services. These services are called a web service, which allows applications to be interoperable. Different SOA tools and methods for modernization and integrating legacy applications will be discussed below.

2.7 Web Services

"A web service is a network accessible interface to application functionality, built using standard Internet technologies" (Kuntal, 2011). Web services are the interoperable services that are available over the internet, uses standardized XML messaging system and are not tied to any operating system or programming language (Tutorial Point, 2010). Web services are standardized ways of transferring data and call functions across the different network, programming language, network and operating system. Web services are the applications components that communicate using open protocols. However, web services are self-contained and self-describing application that can be discovered through the use of UDDI (Nichol, 2008). They can be used by other applications and they use basic XML-PRC and HTTP protocols. A web service is a platform that uses SOAP, WSDL and UDDI elements for the communication with other applications (Nichol, 2008). The use of web service is to connect existing applications and re-use application components

Web services use open, XML-based standards and transport protocols to exchange data with clients. They provide a standard means of inter-operating between different software applications, running on a variety of platforms and frameworks (David, 2004).

Web Services are the services that are interoperable and they communicate over the network regardless of the language and operating system they resides on. These services can be invoked and respond over Internet (Somalenge, 2010). Since the different applications had a problem to interact with each other due to the use different programming languages, form of communication and operating system. These applications had a barrier of understanding one-another. To re-write the applications is expensive and time consuming, hence web services became a solution to that issue. The main objective of the web services is to allow services to be interoperable. Now the legacy application can interact with the modern application via web services without any hindering. The legacy application can expose each application as a web services with a detailed methods and functions. There are several alternatives for the messaging system, one can use XML-RPC or SOAP protocol for messaging, both of which will be described later in this chapter.

2.7.1 XML-RPC

XML-RPC is a language that can be used on a different platform and environment but it still expresses the complex message (Laurent, et al., 2001). XML-RPC is a Remote Procedure Calling protocol that works over the Internet. XML-RPC is a simple protocol that uses XML messages to perform RPC's. Requests are encoded in XML and sent via HTTP POST. XML responses are embedded in the body of the HTTP response. Since XML-RPC is platform-independent, it allows diverse applications to communicate (Josuttis, 2007). It is the easiest way to start web services. However, unlike SOAP; XML-RPC has no corresponding service description grammar. It is a layer that is responsible for encoding messages in a common XML format so that messages can be understood at either end.

An XML-RPC message is an HTTP-POST request. The XML-RPC API allows developers to make XML-RPC calls to the IP Board for information. XML-RPC is a popular and robust method that is used for remote API calls, however XML-RPC is still utilized and IP Board provides a robust library to handle dealing with XML-RPC communications (Laurent, et al., 2001). IP Board uses this library when handling communication with a remote IP. XML-RPC library is a tool that can be utilized to facilitate calls to and reading response from XML-RPC. XML provides a language which can be used between different platforms and programming languages, and still express complex messages and functions.

2.7.2 WSDL

WSDL is an XML grammar for specifying a public interface for a web service (Cerami, 2007). This public interface includes information on all the public available functions, data type information for all XML messages, binding information about the specific transport protocol to be used, and address information for locating the specified service. WSDL is not necessarily tied to a specific XML messaging system, but it includes built-in extensions for describing SOAP services (Cerami, 2007). Using WSDL, a client can locate a web service and invoke any of the public available functions. With WSDL tools, this process can be entirely automated, enabling applications to easily integrate new services with little or no manual code. It gives explicit details of a web service to a client.

2.7.3 SOAP

SOAP is lightweight protocol that exchanges data between peers in a decentralized, distributed environment using XML (Gudgin, et al. 2007). It gives a mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. SOAP is made up of three parts which are SOAP envelope, SOAP encoding rule and SOAP RPC presentation. The major goal for SOAP is simplicity and extensibility. SOAP provides a flexible mechanism for extending a message in a decentralized and modular way without prior knowledge between the communicating parties. One of the design goals of SOAP is to encapsulate and exchange RPC calls using the extensibility and flexibility of XML (Gudgin, et al. 2007).

2.7.3.1 Nu-SOAP

Nu-Soap is the library used in this project to write web services. Nu-SOAP is a rewrite of SOAPx4 that is provided by Nu-Sphere as a group of PHP classes that allows developers to create and consume SOAP web services (Crugnalo, 2006). Nu-SOAP is a good choice for creating and consuming PHP SOAP services. It delivers a complete SOAP implementation for PHP, without relying on any extra PHP extensions, which makes it easy to use. It comes under the LGPL.

Nu-Soap does not require any PHP additional classes. Nu-SOAP is a technology that makes a suitable platform for PHP developer to create web service platforms. SOAPx4 library was used to write SOAP client and SOAP servers and consume SOAP web services; however that generated a problem because developers had to write an overall WSDL document from scratch (Crugnalo, 2006). With Nu-SOAP the WSDL document is automatically created when creating soap a server portal.

All the above mentioned tools are essential for modernizing applications and required for integrating existing applications into SOA middleware. For SOA middleware to smoothly integrate legacy systems, all the mentioned protocols and tools are required. To avoid failure of the integration project and to use certain tools and protocol there is a need to know and understand the SOA middleware that is being used, and how it exposes its interfaces.

2.7.4 REST Web services

REST is a Resource Oriented Architecture designed to leverage the existing HTTP methods to create simpler web services (Aaron, 2008). Rest uses standardized features such as URIs, HTTP protocols, and HTTP status code to design desired web service. REST is more flexible and addresses advanced quality of services requirements. REST is well designed for basic and ad hoc integration of applications.

2.8 SOA Middleware

Middleware is computer software that is designed to automatically carry out a logic operation and provide services to other software applications (Bradley, 2008). SOA middleware is a platform that allows the integration of internal systems and legacy applications, to reduce the complexity of web applications, and cost of maintenance for these services (Makamba and Thinyane, 2011). It is also known as a platform that acts as a plumber or glue between different software components (James, 2012). It is software that lies between the operating system and service applications; it resides on top of the operating system and communication protocols. It connects disparate applications and enables web server to display dynamic web pages based of the client request form. The development of middleware is stimulated by the enormous growth of stand-alone applications. Companies and organizations are now building enterprise-wide information systems by integrating previously independent applications, together with new developments (Schneider, 2003). This integration process has to deal with legacy applications. A legacy application can only be used through its specific interface, and cannot be modified. In many cases, the cost of rewriting a legacy application would be prohibitive. Middleware is there to help to optimize the functionality of stand-alone application, as it connects these applications it allows them to collaborate in the same environment. It makes things easier for developers, when they want to enhance the usability and functionality of the applications. It provides uniform, standard, high-level interfaces to the application developers and integrators, so that applications can be easily composed, re-used, ported, and to inter-operate. It supplies a set of common services to perform various general purpose functions, in order to avoid duplicating efforts and to facilitate collaboration between applications.

The role of SOA middleware is to make application development easier, improve functionalities and usability of legacy applications by providing common programming abstractions, masking the heterogeneity and by hiding low-level programming details.

2.8.1 TeleWeaver SOA Middleware

TeleWeaver is a lightweight, custom built OSGI container which supports online and off-line services, targeted at MRA’s in Southern Africa (ReedHouseSystem, 2011). TeleWeaver is an optimized OSGI container, which is built using Equinox OSGI implementation, and presents benefits of ESB architecture to ICTD environment. It is a customized OSGI container that is developed to fit in the context ICT4D environment, also to help MRA's by allowing integration of legacy applications that were developed for MRA's to be effective and sustainable. Its conceptualization is the outcome of the work done in SLL, which furthered inefficiency and unsuitability of stand-alone applications. TeleWeaver platform improves legacy applications that have been developed for marginalized communities such as Dwesa by allowing the integration of these applications into it (ReedHouseSystem, 2011). The TeleWeaver middleware is a very robust and low-maintenance architecture; at the same time it allows external developers to deploy their software extensions to the platform. The TeleWeaver middleware has been developed by ReedHouseSystem (RHS), an ICTD software company based in Graham’s town and affiliated with the Siyakhula Living Lab. It is the middleware that comes with service adapters and support rural entrepreneurs (ReedHouseSystem, 2011). TeleWeaver is an open source technology that offers blueprints for services. It is poised to become a next generation enabler for rural tele-centre and access node (ReedHouseSystem, 2011).

TeleWeaver allows different applications to be re-usable, interoperable and integrated, also to optimize the legacy applications. For the integration of the legacy e-Services TeleWeaver is essential, as it allows applications to be easily accessible and maintainable. It speed up project implementation and reduces the total cost of ownership for projects, by simplifying the transition from point to point and increase architecture business process changes. Deploying a new application or process utilizing SOA constructs and making its resources available through services requires essential perception of SOA principles. Most of the applications that were developed for SLL are stand-alone applications that require high maintenance cost, and present inefficiency, TeleWeaver is a solution to the above mentioned issues. TeleWeaver uses different tools, such as Java EE, OSGI container, Spring-Framework, Apache-CXF, for development and it can integrate with any other technology such as PHP, Python, and ASP (Reed House System, 2011).

TeleWeaver has an OSGI container that enables software to run on a Java platform. Through OSGI container, TeleWeaver is hybrid architecture that offers several services. It uses series of technology for the development environment, such as IDE eclipse for developing application, Maven, POM, and Archetype as a Building tools. It uses GWT as an interface, OSGI container, and subversion as a repository. TeleWeaver has GWT that communicate with API of the services within the OSGI container. The IDE eclipse is a developing tool that is used to develop applications, and bundles in TeleWeaver. Maven is a building tool that is used to clean the code and detect the debug, without a developer to look for error in a code, it speeds up the process of developing application by presenting where the error is allocated and what is required, and how to fix it.

POM is the fundamental unit of work in Maven (Tutorial Point, 2010). It is an XML file that contains information about the project and configuration details used by Maven to build the project. The POM file contains the dependences of the applications. POM contains important goals for the Maven project; it contains project dependencies, plug-ins that can be executed. It also embeds information such as group-id, artifant_Id, project version, and URL description and developers details.

Archetype it is a generic tool that allows the inheritance of components. it is a tool or temple that allows developers to quicken the development, as it inherits the common functions of modules. This tool minimizes the process of writing the same code for common components, it allows developers to write a code for different modules, and inherit the code for common functions, and hence it speeds up the development process.

It uses several tools for the deployment environment, tools such as, JRE, Equinox as a services environment, Data transaction layer, Validation layer and security layer. JRE is being used as a service environment, while the Equinox has been utilized as a run time framework that allows developers to implement services as a set of bundles. Security is use to check the users' credentials and validation is for checking authentication and authorization of the end users.

TeleWeaver uses Google Web Toolkit (GWT) as a user interface. This interface typically eliminates a bad design pattern, by separating business logic from presentation logic. GWT is a front end module in TeleWeaver that allows a client to view the service functionality. It is a tool that allows developers to create user friendly interfaces, without writing an HTML code. GWT is selected due to its flexibility and usability. It is a tool that allows components to have API and different implementation code but to have a single view interface.

2.8.1.1 OSGI

Open Service Gateway Initiative (OSGI) is a mature technology with a broad range of adoption (Tim, 2012). It is essential for managing transaction bundles and for remote services. OSGI is used for various reasons but its main focus is modularity. Big systems are difficult to understand and hard to maintain because of different relationships between complex components, OSGI comes as a solution in this issue. Most of the applications are complex and have more external dependences, an OSGI has a class-loader for library classes and it also allows developers to number the versions of library that are used. OSGI is mostly used for bundles; its bundles are like JARs with a better isolation. It is also used for a well design of a code with a good exhibit of modularity properties, for simple reuse and easily switches to implementation. The modularity properties rely well on classes being cohesive and loosely coupled. For OSGI bundles to work there must be cohesive and loosely coupled. To design a distinct bundle a bundle manifest is an excellent guide.

OSGI services are actually less verbose and much faster. Still, much of the service-oriented mindset applies to OSGI modularity. The services are agents of modularity between systems. The difference is that Web services generally allow communication between systems distributed over some distance via the Internet, whereas in OSGI local services connect subsystems (modules) that interact to form complicated applications. These local services are extremely important in the OSGI model of modularity. These services, when done properly, are what allow truly loose coupling between modules. It is extremely important to ensure that each module is blind to the internal operations of other modules, so that they cannot have any dependencies but express the output. It makes the modules inter-changeable parts.

OSGI is a container that allows different components to reside on it; it allows modules or components to interchange information. It is services platform and modules system for Java programming language that implements a complete dynamic component model (Tim, 2012). These components are separated and they do not know each other's internal operation, and they do not know that they are residing on the same container. They also do not know how each component operates; they communicate when they have to get method from other component, or when it is necessary. So if one component is not working it will not affect the other components, which is the beauty of using the OSGI container. By having stand-alone components we avoid the mess that happened on the stack applications, whereby if one component is dead the whole system will not function. By having separated components it become easier for developers to add other components within the container, within a small amount of time. Unlike when dealing with stack applications whereby a developer has to understand the whole application architecture in order to add other services on the application. OSGI minimizes the time of coding, of understanding each application's architecture and it increases the productivity of the company by allowing separated components within the container. The components within the container can be called remotely, can be installed, stopped, removed and resumed without requesting a reboot of a system. Any framework that implements OSGI provides an environment for modularization of applications into bundles.

OSGI contains bundles; a bundle is a group of Java classes and additional resources equipped with a detailed manifest file on all its contents and also additional services needed to give the included group of Java classes' sophisticated behaviors, to the extent of deeming the entire aggregate a component (Jack, 2007). Bundles are also known as normal jar files with an extra manifest of headers. OSGI has specified many services and these services have been specified in a Java interface, bundle can implement these interfaces and register services with a services registry. A service registry is what allows a bundle to detect the new additional service or remove services.

DOSGI

Dynamic Open Services Gateway Initiative (DOSGI) is a service invoker that exposes all the components that reside within the container and eliminate the challenges that each third party had to face when having to access the components inside the container. TeleWeaver uses DOSGI as a center of communication within the container. DOSGI removes the governance of each application by exposing all the essential details for each component. It is also a framework that enables deployment of web services as OSGI bundles, which in turn enables deployment and supports multiple versions of a web service. DOSGI communicates with all the components within the container, making it easier for the third party applications to access all the components in the container. Unlike previously, whereby the third-party had to know the setup rules of communicating such a name, package, URL and the list of methods and classes of each components when it want to communicate with it. It became a challenge for a third- party application when it wanted to access different components as it has to know all the above mentioned protocols when it wants to access components. The components within the container interact and they have different names, packages, methods, URL and classes. One had to know all these contexts of components when wanting to access it. This has become a huge problem when one wants to access the different components. With the help of DOSGI the third-party has to know only a URL of the service invoker (DOSGI) and package of the component that want to communicate with. Now the third-party or end user has to pass a method that it wants to be retrieved to a service invoker, the services invoker checks which method is contained by which components and gets a response from that component and passes it to a third- party. Instead of the user to talk to each module, it communicates with DOSGI to retrieve the information that it wants. DOSGI abstracts the important content of each component from xml file that is in a drop-box within the container, that xml file in the drop-box inside the container specifies the name, package, methods and classes of the component.

The drop-box is the box where each component has to drop its xml file so that the invoker can see what name of the component and which classes and methods are exposed by it. It is designed to reside within the OSGI container.

DOSGI uses the Apache CXF the Apache CXF Distributed OSGI sub-project provides the Reference Implementation of the Distribution Provider component of the OSGI Remote Services Specification. It implements the Remote Services functionality using Web Services, leveraging SOAP over HTTP and exposing the Service over a WSDL contract. Starting from versions, URL, Java interfaces can also be exposed and consumed as Restful JAX-RS services (Apache, 2010).

SOA middleware will enable applications to be re-usable and also more usable through using Single Sing On (SSO) functionality. There is a lot that has been done by ICT4D context under SLL. The following paragraph will explain more about libraries that were used to integrate legacy applications into TeleWeaver, ICT for community development and the aim of it and how ICT helps in improving the lives of MRA's, how it helps in overcoming the challenges of MRAs and the benefits of using ICT4D in SLL.

2.8.1.2 TeleWeaver Middleware and Libraries employed for integration system

TeleWeaver is the main platform that is used for integrating these legacy services. TeleWeaver exposes various interfaces for the integration of applications. This is to enable the easy integration application process. To integrate disparate services, certain mechanisms have to be used. For these two legacy applications to be integrated into TeleWeaver, certain tools such as Web Services and APIs are required to be used, to enable easy integration procedure.

Since web services make it possible for diverse applications to discover each other and exchange data seamlessly via the Internet. In this project RESTful web service has been used to exchange data.

The main goal of using this web service architecture is to allow administrators such as art and craft designers and school teachers, to upload their products, information on the applications through TeleWeaver. The administrators will be able to create, edit and delete their personal accounts, information about products, and products through a RESTful web service interface. RESTful web services leverage URIs to allow clients to have an access to service's resources (Aaron, 2008). REST allows developers to make more dynamic services without affecting client links to the resource.

Google Web Toolkit.

Google Web Toolkit (GWT) is a component that resides within TeleWeaver. It is a core application view component. It is also known as a front-end tool. This is a component that allows an administrator to view services that are available in TeleWeaver, to upload products and information to the applications. The beauty of using this tool is that it separates the business logic from the presentation logic, hence it become easier for developers to maintain the system. The view application components know nothing about the implementation of the core service components in the system. When an administrator wants to view the certain components, GWT sends a message to API. It's the API that communicates with the implementation methods of the application and retrieves back information required by the administrator. In this context GWT separates pattern. It optimizes the system functionalities and leverage's developers operation. The only aspect that GWT understands is that there is an API that can communicate with the implementation method of the services. Its main goal is to present the service that is available, and to allow administrators to upload information and the product to applications.

2.9 Application Programming Interface

API is a main core service component that links the third party application with TeleWeaver middleware. It is the major component that allows communication between Middleware components and the third party components. The contextualization of API in this system is to allow components on third-party services to communicate with the middleware. This API is the main mechanism that allows integration of application. It allows modules within legacy applications to communicate with the API server. The API server will interact with the service client and give back the required information. The API server is the mechanism which interacts with the various modules within applications. It then gives the requested information back to the client. When the administrator seeks to upload information to application, she/he will upload it using GWT components. The GWT will initialize the communication with the API, and then API will transfer the uploaded information to the intended application. Then the information will appear on the legacy application.

The client sitting on the other side of application will retrieve the uploaded information. He will able to view the new product that is uploaded by the art and craft entrepreneurs. Hence this system integration is assumed to make the legacy applications flexible and effective.

These above mentioned tools are here to make sure the system works accurately. These tools enable the user to interact with the system. These are here to make system integration easy and efficient. With these tools integration of legacy applications is easy, fast, and effective. However, we will need to see how the user interacts with the system. Later in this chapter there will be a demonstration of how user communicates with the system. The role of the user will be elucidated explicitly. All the above mentioned libraries are the libraries that TeleWeaver uses to integrate legacy applications.

2.10 Application Integration

Application integration is typically a complex and costly process, in which different techniques are required for different technologies (Karen, J.B.2011). Many companies have different applications that are built over different languages which uses different technology that run on different platforms and have similar functionality such as log in, profile and database, these applications need to be integrated into a platform to reduce the redundancy of data. Application integration is an important aspect in software industry. Many companies or industries have experienced problems with multiple stand-alone applications as they are difficult to modify, expensive and they consume lot of time. These stand-alone applications can be integrated into SOA platforms, for them to be effective, valuable and efficient.

Application integration has long been seen as a stumbling block to any new software implementation; however, new tools are making the process easier and spurring a faster return on investment (Karen, 2011).

In many businesses data should flow seamlessly from application to application, and most users are dealing with soiled environments where data stays within a specific application (Karen, 2011). This can be fixed by integrating applications, unfortunately without integration, end users are stuck manually entering data into multiple systems and applications, a process that is often haphazard, ineffective and plagued with errors. As a result, business decision-makers do not have the visibility, agility and flexibility needed to make serious and often critical business decisions, which decrease productivity in the company.

The integration of application into a TeleWeaver middleware presents flexibility to the services as well as the extensibility. Using SOA integration of legacy applications will leverage the existing investments. SOA and TeleWeaver provide a mechanism to front- end system without a modification on the existing application. Integrating legacy applications increases the flexibility of connecting clients with the e-Services. It allows end users to have more access to different services in a high speed and flexible manner that result in socio-economic development of the community. Integration of legacy applications into TeleWeaver offers a visibility of services as well as easy access to the e-services. The integration of e-Commerce and e-Learning applications into the TeleWeaver SOA middleware increase the development productivity, also to reduce the processing time for the applications and provide user friendly applications. Application integration allows applications to be flexible and interoperable.

2.10.1 Advantages and benefits of application integration

One of the main benefits of integrating applications is that it eliminates the cost of maintaining stand-alone applications. It gives a value to an existing application. It improves the performance of legacy application as it integrates these applications into new innovative environments. It allows application to be effective at a lower cost. It eliminates the entering of the same data on disparate application and eliminates the consumption of time of stand-alone systems. It cuts development cost of applications and makes application to be fast and effective. It allows developers concentrate less on developing a code, rather than allocating a routine code task. It makes changes easily with no internal disruption. It provides more delivery channel and allows modification of applications.

Integrating applications into an SOA middleware presents a number of advantages. Application integration improve the time and monetary costs of maintaining legacy systems as it allows easy integration of third party tools that help the entire system to be more malleable and dynamic. Integrating legacy application exhibits positive impact on applications. It leverages the existing system by optimizing the functionalities and sustainability of these applications. Integrating legacy applications provides fast delivery flexible and reliable access to processes and information regardless of the platform (Karen, 2011). Lot of time have been spent on IT budget, for maintain the existing application. According to Gartner Group amount of 60% and 80% of IT budget is spent simply on maintaining existing mainframe systems and applications besides being expensive, a lot of time has to be spent looking at each application architecture when something has to be modified(Bradley, 2009).

Integrating legacy applications provides fast delivery, flexible and reliable access to processes and information regardless of platform or data access. Gartner Group recently estimated that between 60% and 80% of IT budget is spent simply on maintaining existing mainframe systems and applications besides being expensive, a lot of time has to be spent looking at each application architecture when something has to be modified (Gartner, 2008).

Application integration ameliorates legacy application effectiveness by allowing real time communication between services. Another advantage of integration legacy applications is that applications become easy to modify, flexible and sustainable. Applications become efficient and effective. Integrating legacy applications improve the effectiveness of the legacy applications and preserve the value for these applications. It make applications easy to use by end users (Karen, 2011), and allows applications to meet the new customer needs, or business requirement. Integrating applications makes a rapid productivity for a company, as it makes a work easier for developers, it enhances developer’s productivity. It provides a real time communication between multiple services and clients.

“Integrating existing applications will save the budget in organizations or companies as it reduces the cost of maintenance for these applications and it makes user friendly interfaces for each application” (Karen, 2011). Having applications that are integrated in a single environment will provide an easy access to the applications and flexibility, especially for MRAs environment, as SSO will be provided to make an easy access across multiple applications.

To support the expansion of services and improve functionalities and operations, ICT4D decided to standardize its software portfolio, while at the same time integrating applications to gain a clearer view into its revenue pipeline. The ability to utilize AIA was a key deciding factor, since it would streamline the process and allow data sharing and integration between the disparate applications (Rafael, 2007).

2.10.2 Ways of Integrating Applications

There are various ways of integrating applications, Manual application integration, Semi-automated application integration and fully automated application integration (Microsoft, 2009). One of the effective ways of integrating legacy applications is to preserve and update the critical business processes by developing an incremental application integration road map. Road map helps by reducing the chances of unsuccessful application integration project. It provides criteria that must be   followed to successful integrate legacy applications. Criteria involve certain steps which includes, Application assessment, Target definition, Technical understanding of each application and certain recommendations. This process with these distinct steps leads to a decision point of accurate integrating existing services. This accurately determines how to best integrate legacy applications.

The first important step is Define target; to define what structure the application portfolio should take following the modernization effort (Dickerson, 2003). Give an attention to resource constraint issues by tracking the effort required to complete each portion of integration. Application Assessment: to analyze your existing applications, preferably in an automated fashion, to determine their complexity and structure. This assessment can help to set the criteria for identifying and isolating your business rules.

Technical Understanding of application is essential; to better understand the nature and usage of the applications from a user perspective and developer's perspective. The final step is a Road Map; a synthesis of knowledge gained during the earlier steps, combined with detailed metrics on the applications provides pipelines of how to proceed with your effort (Dickerson, C. 2003). The above mentioned criteria of integrating application are necessary when integrating legacy applications, as they determine accuracy of method that will be used. They are the primary step of integrating applications. The method of integrating must proceed after these criteria. For ICT4D intervention it is important to draw out these approaches to integrate numerous services that developed for MRAs, to avoid the risks and failures of the integration project.

Different enterprise environment has been integrating legacy applications use different methods and technology. Below are the ways that enterprise environment has been used to integrate their legacy services.

2.10.2.1 Manual Application Integration

Manual application integration requires people to act as the interfaces between applications and enable the integration between them. People may enter the same information into multiple systems and read information from those systems to respond to customer requests. In other cases, a person may need to read customer information from one database, and then re-enter it into another database used for another purpose (Microsoft, 2003). Manual application integration becomes more complex, however, when your organization becomes more complex, and can lead to inaccuracies in data. As the amount and complexity of your data increases as the number of applications increase, you will require more and more people to maintain such an environment. An environment that relies heavily on manual integration is generally very inefficient, and does not grow as easily as other application integration processes.

2.10.2.2 Semi-Automated Application Integration

Semi-automated application integration combines individual software components and test them as a group (Microsoft, 2003). Semi-automated application integration generally requires more technology investment. In this case, all of the steps before and after managerial approval may be automated, but a person is required in the middle of the process. In other cases, human intervention may be needed to transform data that is required in another system.

2.10.2.3 Fully Automated Application Integration

Fully automated application integration is another way of integrating application, by exterminating people from the business process, although they are required to maintain the solution (Microsoft, 2003). This type of integration consists of applications communicating through a series of interfaces and adapters. Although fully automated application integration removes the dependency on people, such systems can be more expensive to implement and may not be practical for many business problems (Microsoft, 2003). In many cases, you will still require people to make business decisions, and it is often more efficient to have a person control a technical process as well. For these reasons, you should decide where fully automated application integration is appropriate on a case-by-case basis. This system is subjected to create errors and it is expensive.

A lot of challenges were encountered during the integration process; these challenges will be discussed in the paragraph below.

2.10.3 Challenges of application integration

Application integration has its own challenges which include controlling and coupling of data formats and concurrence. Some of the most significant technical challenges of designing an application integration environment involve identifying the technical needs for your solution and determining the combination of products and services that will provide for those needs. Because this segment of the market is still evolving rapidly, you should pay particular attention to the maturity of integration technologies, methodologies, and standards. Organizational issues in application integration can be particularly tricky when you are working in a decentralized enterprise or in a business-to-business (B2B) scenario. However, these scenarios often have the greatest need for an effective application integration environment. Therefore, it is particularly important to resolve any organizational issues earlier on, so that you can focus on resolving the technical challenges of application integration.

One of the challenges of integrating applications is that it is rare to find one application that provides all of its software needs (Karen, 2011). There are many reasons that application integration is challenging. The first is that, it is rare for a company to have a single application that provides all its software. Connecting various systems, companies need comprehensive integration capabilities that can deliver business value at a lower total cost of ownership. Another challenge is that a developer will not know which application component is important to the user, a lot of time might be spend on integrating components that are not of use to the companies. That is the biggest problem with all of these disparate challenges.

IT developers or managers in enterprise environment, overwhelmed with the prospect of having to mitigate these problems, may choose to do nothing, sticking with the status quo, or will integrate applications on an ad-hoc basis using a variety of integration methodologies (Karen, 2011). The challenge is the complexity of legacy applications. For application integration to be effective a lot of tools and technologies are required, to deliver a comprehensive, sustainable integration solution and simplify the integration experience for customers.

Another challenge for application integration is that most people do not see the big picture when it comes to integration. They see the immediate applications that need to be brought together but they do not understand that good integration requires a very specific, calculated approach. Companies should match the complexity of the solution with the complexity of a problem that needs to be solved.

The developer must consider which style of integration they need to use to solve their business requirement, is the integration happening at the presentation layer, the business process layer, or the data layer. Companies must have modules that orchestrate the specific requirement for integration. That allows developer to defined services as re-usable components for future integrations. Most of the challenges that developers encounter during the integration of legacy applications is that, it is time consuming to integrate these application due to the complexity of these applications. The last mile of integration which is a final delivery of connectivity, some refer to it as a reliable figure, since it is an SOA based way of integrating legacy applications with the modern architecture and supports business initiatives (Seagull, 2011). Below is discussion of how other industries have used middleware platform to integrate their legacy applications. The following paragraphs will explain what kind of middleware that were developed, and their functionalities and how they assist in the integration of legacy applications

2.11 Related Work

Integration of services has been done by different companies but the integration of e-Learning and e-Commerce has not been put into consideration. There are a lot of middleware that have been developed to integrate services. The following section will give more details about the middleware that is used in enterprise environment.

2.11.1 The iWay middleware

The iWay SOA middleware is there to overcome the challenges relating to SOA (Freival, 2006). It isolates business requirements from changes in infrastructure and liberated business to better align IT with business goals. With iWay SOA middleware, services can be introduced while allowing the experts to retain control of the underlining applications and providing end users with the most user-friendly access to resources. It provides a best interoperability for more infrastructure components than any other company in the market today.

The iWay makes all aspects of your existing environment work in concert (Freival, 2006). This includes non-standard legacy applications that are typically ignored by other SOA vendors. The iWay middleware have suites, the suite of iWay product enables business to expose functionalities and information resources from a wide variety of perspective. It provides a true agility and flexibility for the enterprise. Without writing a code, services and interfaces can be rapidly developed and deployed so that services consumers can use the automated business process.

The iWay SOA middleware also have services managers that are considered as an open transport ESB to provide a single platform for SOA, EDA and extend to B2B services design development. An iWay SOA middleware can run stand-alone or can be directly deployed on another vendor’s platform or open source platform. The iWay SOA middleware includes support for a wide variety of existing software so that customers can keep and use any applications and middleware they choose and take advantage of any existing standard (Freival, 2006). It enables rapid incremental business process changes.

2.11.2 SAP NetWeaver

Sap NetWeaver is another middleware that focuses on integrating the services. SAP NetWeaver middleware makes it possible to integrate heterogeneous IT landscapes and move forward to SOA roadmap, as it provides a functionality and scalability needed to integrate the legacy world with SAP on an enterprise level (Laurent, 2010). SAP NetWeaver process a bulk of messages in one service call (mapping and routing). This middleware reduces context switches, enables mass operations on the database and it is useful for asynchronous scenarios.

2.12 Conclusion

This chapter has reviewed all the literature that is related to the integration of legacy application into the middleware platform. This is to achieve the second objective of this research, which is stated in chapter one of this thesis. The technologies that are discussed in this chapter are vital, as they describe the advantages and how to integrate these legacy applications into SOA middleware. These technologies give a clear road map on integrating the legacy application. The next chapter discusses more about methodologies that were used to modernize and integrate legacy applications.

3. Chapter 3: Methodology Perspective on Integrating Legacy Applications into SOA middleware and System Requirements

In this thesis there are specific methodologies that were used to explore the application integration process; these methodologies are discussed in the following sections. Each section followed its context and cooperating methodologies. The main objective of this chapter is to explore the methods of integrating legacy services into SOA middleware.

This chapter gives the methodology perspective on how to integrate existing application into TeleWeaver middleware under SLL intervention in order to achieve the stated objectives in Chapter One. It also gives an overview of system requirements and the tools required for the implementation of this project. It demonstrates more on the methods of integrating existing application under SLL context. This will highlight explicit details of the modules that are used to develop the integration application system. The description of the system front and back ends will also be discussed later in this chapter as well as system entities and the user role. This chapter illustrates how this research aims to achieve the stated objectives.

3.1 Introduction

This chapter discusses the methodologies used to integrate legacy application into SOA middleware. However we discovered that there are several technologies and methods that are available for the development of integration portals. Several methods have been used in the enterprises environment for integrating legacy applications, therefore this chapter will reveal the methods used for integrating existing applications under SLL context.

The system requirements will be discussed later in this chapter. System analysis and requirements elicitation will be discussed through various methods. Functional and non-functional requirements of the system will be discussed later in this section, the user interaction with the system, and various roles of the user with the system will be elaborated.

3.2 System requirements

To integrate legacy applications, certain tools and methods are essential requirements. This is to avoid the failure of the integration project. For this project the requirements of the system were met through the use of different methodologies: qualitative research, interviewing end user, literature review, the study and understanding of integration and application modernization techniques.

The first scenario was based on the field visit to Dwesa, to interview the community and school teachers about integration of these services into SOA middleware. People from different backgrounds were interviewed. The findings from the interviews confirm that having e-Services integrated into Middleware will make easy access to the different application and that will make it easier for art and craft people to upload their products on Os-Commerce application using TeleWeaver. The crafters complained of slow business flow. The system proposed in this research will accelerate the flow of the business for art and craft entrepreneurs. It will also make it easier for school teacher to update e-Learning application all the time. This ensures that integrating legacy applications into TeleWeaver SOA middleware will improve the efficiency, sustainability and flexibility of these applications. It also enhances the accessibility to these applications.

However, this application integration system will not only improve sustainability and efficiency of these services, it will also improve the education system at Dwesa. The administrator or teachers can upload information on TeleWeaver platform where ever they are and the information appears on the e-Learning application. Now the students can view the new updated information whenever they want, they no longer depend on the teachers to arrive in order view the current information. This system is vital to learners, community, teachers and art and craft entrepreneurs, as it accelerates the business for the community also equip student with the information they need.

The interviews established that the Dwesa community is struggling to remember each password for the different services due to different issues. By having applications integrated into TeleWeaver middleware, will enable easy access for them as it allows Single Sing On (SSO). The administrators have single credentials for different applications. The administrator enters credentials once and the system automatically allows him or her to access every service that he/she is authorized to access. This is to eliminate the password problems, as this system is design to help rural community. It should be flexible and accessible to them.

It is also discovered that the Dwesa community is struggling to grow their business as the e-Commerce application is difficult to modify and to maintain. However, having these isolated application integrated in single platform, it is easy to maintain and sustain these applications.

The second scenario was to study and understand the TeleWeaver middleware and its functionality. Since these two services have to be integrated into TeleWeaver. It is essential to know the full functionality of this middleware and the interfaces that it exposes and support, also the language that it uses. It is important to know what is contained in the core service components, noncore services component, view layer and data layer and service broker of TeleWeaver.

To understand a layer where integration is going to take place, it is vital to know and fully understand the middleware that these applications are integrated to. The most important part in this integration application project we need to know which type of middleware is TeleWeaver, and the interfaces that are exposing by this middleware, and the way of integrating these two applications. There are different types of middleware, by knowing the type of middleware will enable fast integration process and we will know where which layer these applications will be integrated into.

The literature review has been done to understand the context and to fully understand the existing applications that were developed and how other people integrated their services into the middleware. To study and understand the e-Commerce and e-Learning applications and how they were developed which language they are using so we could find the way of integrating them in the middleware. Knowing every module of these applications will make integration process very easy.

It is also important to understand the requirements of the application integration project and ways of integrating the legacy applications into SOA platform. However, the most important phenomenon is to understand how are these application are integrated. This proves that methodology is important phenomenon in the integration project, this reduces the chance of an unsuccessful project and the amount of time spent on integrating existing applications.

3.3 Methodology use to achieve integration project

Requirement is an operation that a computer application must do for its users (Daryl, 2004). It is a specific function or principle that the system must provide in order for it to merit. The requirement focuses more on how system should operate or interact with the user, rather than focusing on what are the steps that the system should follow do carry out the operations.

For this project specific system requirements have been considered to avoid the failure of this project. The following section discusses the methods that were used to meet the objectives of this study.

3.3.1 Requirement Elicitation

Requirement elicitation is the research methodology that was adopted in this research to achieve, the sub-objective 2 which is mentioned in Chapter One, and to understand what the community wants. This was done through the trips that the researcher has taken to the Dwesa community to communicate with the community and other stakeholders. For this project the developers need to take the initiative when developing a system for the users. Since users they find it hard to articulate what they want. Then the first scope of the requirements elicitation is to defining the problem, which is to integrate legacy applications into SOA middleware.

These applications are ineffective and unsustainable. It is important to note that requirement elicitation does not depend only on certain stakeholders but it involves everyone on the development of the project. In this case the requirement elicitation depends on the information gathered from the administrators of the system, such as art and craft people, the developers and the analyst of the system. Despite this importance of understanding the problem, this research has measured the effectiveness of various and essential requirements elicitation techniques.

Requirement elicitation is a non-trivial procedure; hence the techniques have to be used in this project. We cannot be sure of getting all the information that we want system to do by just asking users, interview and other procedures or techniques must the conducted. Requirement elicitation is not only a method that is conducted, there are also other methods that we conducted to achieve the objectives of this study.

3.3.1.1 Interviews

The interview method was used in this research to understand what the community wants. During regular visits to Dwesa to conduct computer literacy lessons to students, teachers and other community members, informal interviews were conducted to elicit system requirements based on end users’ perspectives. This was a vital exercise which enables the researcher to collect information from the users' perspectives and also to investigate the challenges that rural ICT users’ are facing. This enabled the developer to make changes on prototyping. These informal interviews were primarily group discussions with teachers, headmasters and community member. Questions asked in these informal interviews were essentially about legacy application integration, legacy application management and functionality of the legacy services, modernization, sustainability and effectiveness of existing systems. During the discussions it was discovered that legacy application were not effective and they are unsustainable. In such discussions issues related to difficulty of modifying and maintaining existing applications and system in-flexibility were raised.

3.3.1.2 Observation

Before the integration took place, we needed to understand and identify the integration technique that can be used to integrate these legacy application; however it is essential to observe how legacy application works before integrating them into middleware, so to see the improvement and the effectiveness of the integration process. The researcher conducted meetings with crafters and school teachers, and the findings show that these legacy applications are ineffective. The crafters and school teachers are the people who use these applications.

It was observed that legacy applications are not easy to maintain, not effective and time consuming when system has to be updated. It is observed that whenever the application has to be modified and updated, a person with a good programming background is required. When the art craft entrepreneurs want to upload product on the system, someone with a good background in computers is needed. This makes the applications to be unsustainable and not effective, hence the e-Commerce business is slow, and because of the time one has to take to upload the product on the application. For e-Learning is the same protocol as well, for the school teachers to upload information on the system, someone with the better understanding of the e-Learning application is required. This makes a slow development on the education system of the MRA areas.

It was observed that maintaining legacy applications is very expensive and time consuming. Therefore, it is important for the existing applications to be integrated into SOA platforms, to improve the effectiveness of these applications also to allow these services to be reusable, cost-effective and flexible while expanding their functionalities. SOA ameliorates the time and monetary costs of maintaining legacy systems as it allows easy integration of third party tools that help the entire system to be more malleable and dynamic. Based on the observations, it is anticipated that the integration of these third-party applications into the SOA middleware will improve e-services development, and provisioning in Marginalized Rural Areas (MRA's) and ICT4D contexts.

3.3.1.3 Brainstorming

Brainstorming is a process for developing creative solutions to problems (Margaret, 2009). Brainstorming is another method that was used to achieve the main objective of this research of integrating legacy applications into SOA middleware. It is also another method that was used to identify the modernization and integration technique. The researchers and stakeholders have brainstormed during the research process and they came up with vital idea of integrating these legacy applications.

Brainstorming involves spontaneous contribution of ideas from all members of the group and it is a creativity technique of generating ideas to solve the problems (Davey, 2008). In this thesis brainstorming was use to gather information on how the legacy system can be improved, what the problem of these legacy applications is.

The brainstorming technique has been used in this research to evaluate the necessary system requirement of this research and it has been done with the variety of stakeholders during the meetings to discover the requirements. Brainstorming is a popular mechanism that is used to find a solution to a particular problem (Davey, 2008). It is of advantage as it brings the diverse experience of all stakeholders in to play during problem solving. It riches ideas explored so that solution can be found for the problem. It helps to have different ideas from different people and different background.

Most of the solutions have been found during the discussion and the way the stakeholders feel about the legacy application has been clarified. Idea reduction is another method that was used to achieve the main objective of this research. As the stakeholders and the developer brainstorm, there important ideas that researcher come up with. However, these ideas need to be analyzed in order to achieve the stated objectives. Idea reduction is a process of analyzing the ideas that were generated during the brainstorming (Michael, 2007). It is useful tool that is used to evaluate the gathered information about the system requirement. As the system is developed for MRAs environment, it is important for stakeholders to be involved in pruning ideas, prioritizing, grouping ideas and define features. In this project idea reduction has been used to analyze the ideas, and to find the way of integrating these applications. Idea reduction is important when designing a system to involve all the idea reduction approaches to have a clear idea of how the system will function, and how the users want it to look like. In this project, using requirement elicitation and its techniques and approaches have helped to design the system, since we are fully aware of what the users desire to see.

3.3.1.4 Prototyping

Prototyping is known as a revolutionary approach to product development that allow developers to design, visualize, and simulate products rapidly and cost-effectively (Autodesk, 2012). Douglas (2000) defined prototyping as a software development life cycle model in which a software prototype is created for the demonstration of a conceptual solution to a given problem. It is a fundamental method that is used by designers to acquire information from stakeholders about the future design of the system.

Prototyping method was also adopted during the course of this research; this was to complete the main objective and sub-objective 2, 5 and 6.

This method helps to deliver quality products faster than ever, create more innovative concepts and engineer more accurate, develop compelling, realistic visualizations to experience products before they are real and perform simulations to optimize designs (Van Buskirk, 2010).

In this methodology the system goes through the cycle of defining system, developing, implementing, testing and refinements. However, prototyping involves series of phases which are the following.

- Defining basic system requirements.
- Developing a working prototype based on the system requirements of the integration project.
- Deployment of the application integration system.
- Testing the system which includes the feedback from the users, developers and the stakeholders.
- Adjusting the system requirements as per client's needs.

The benefits of using this method, it enables usability testing early in the development process. It focuses on content and functionality. It reduces development time, development costs. It exposes the developer's potential future system enhancements. It also facilitates system implementation since users know what to expect. Developers receive quantifiable user feedback. Prototypes make it possible to get a formal approval of the design from both programmers and the client, before you proceed to the development stage. Prototypes comply with the wish to show fast results to the client. It results in higher user satisfaction. In this project prototype has been used to deliver fast results and reduce development cost and time.

Using a prototype model, a fundamental and key functional requirement was provided based on the basic system requirements. This has enabled a researcher to integrate and modernized the legacy applications for the user's satisfaction. This has allowed researchers to experience and get hands on, on the integration approaches. It has allowed the researcher to experience current system functionality on the user's perspective and generate more requirements. As a result prototypes have become a vital tool to identify and determine the system requirements. The requirements elicitation techniques have been used to gather the information for integrating existing applications for MRAs. These techniques deemed to achieve the integration project success. This proves that certain requirements are important if one wants to achieve application integration project. Figure 3.1 shows the steps involved during the development of this project and all stages it went through.

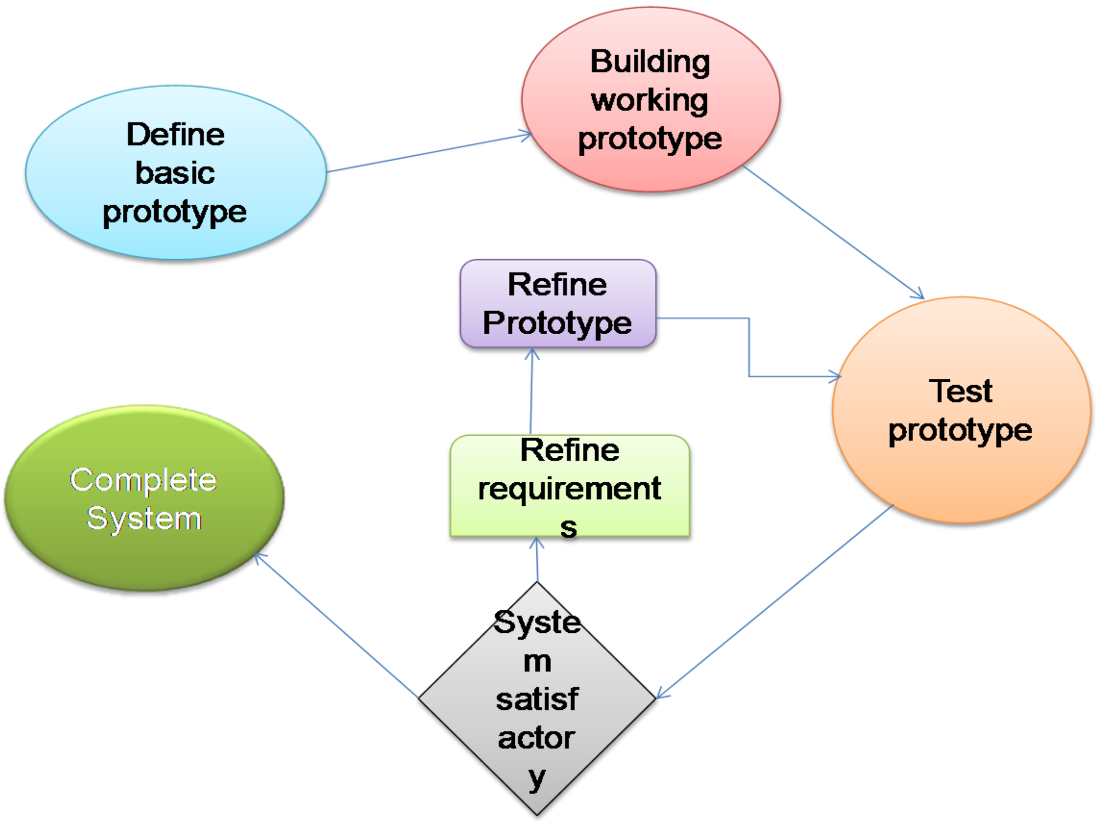


Figure 3.1 Prototype of the system

These are the sequential stages this project went through to carry out results. The first was to define the basic system requirements for this project. It is essential for the developments of the system to understand the basic system requirements. This reduces the development time and cost. It increases the chances of a successful integration project. Having a developed working prototype is also important when working on the integration project.

3.3.1.5 Literature Review

This is another method that was used in this research to achieve the main objective of this research. Using this method researches get to understand and identify the integration and modernization techniques that can be used to integrate legacy applications. Literature review is the first methodology that was applied to achieve stated objectives. This was to gather all the information about application integration also this method has equipped the researcher with essential information on how to integrate legacy applications. This method was used to answer the first research question which includes identifying the advantages of integrating legacy applications into SOA middleware. This was done by reviewing the literature to understand the context of the research, as well as to identify the integration and modernization techniques. Literature review has helped the researcher to identify the best way of integrating legacy applications into SOA middleware.

3.3.1.6 Developing a Working system

Developing a working system is the method this research has adopted to meet the objectives. Through working system this research was able to achieve the main objective of this research also to answer the second research question. This method was used to undertake the integration of related third- party applications into SOA middleware. This method also achieves the main objective of this research which is to integrate e-Commerce and e-Learning application into TeleWeaver middleware. Using this methodology, it is possible to observe if the main objective of the system was achieved and successful.

3.3.1.7 Testing and Evaluation Method

The testing and evaluating method was used to accomplish the last objective of this research. This method was used to test and evaluate the effectiveness of the integration platform. The testing and evaluation method is the main method that answers all the research questions, without this method it is impossible to see if applications are well integrated. To test the system helps a developer to see if the system is working properly and if the system meets all the stated objectives. The core functionality of the system is achieved method when this method is adopted. The functional requirement and functional requirements are achieved through this method.

The above methods were used to achieve the objectives of this project. However, the requirements of the system are also explained in this chapter. The following section discusses the essential requirements for this integration system. It is vital to analyze every requirement of the system before developing the entire system. The following section elucidates the requirement analysis of this system.

3.4 Requirement Analysis

A requirement analysis is the process of determining user expectations for a new or modified system (Margaret, 2007). It involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of unnecessary features and documentation of all aspects of the project development process from start to finish (Margaret, 2007).

Data gathered during the interview with end users, stakeholders and developers shows the lack of effective on the legacy application and unsustainable of the existing applications (Margaret, 2007). It was discovered that legacy application are difficult to maintain and require lot of time when they have to be updated. The developers from SLL are complaining about running legacy application, as they are complex and time consuming.

This research has discovered that integrating these legacy applications into the TeleWeaver middleware will make these legacy applications effective, easy to use and sustainable. The critical components that the system should handle have been discovered which includes communication between TeleWeaver middleware with the legacy application, enabling art and craft entrepreneurs to upload their items on middleware. To allow an administrator to upload the items on e-Commerce platform so that anyone can see the recent uploaded items, by doing that, this will rapidly increase the business for these rural entrepreneurs. This will also accelerate the education system for MRAs, as it improves the education system of those isolated communities. Now the students can access the information at home using Moodle system. This will ease the life of those communities and also satisfy the communities at large. However the system must state the all the requirements including functional and non-functional requirements.

Every integration project must have functional requirements and non-functional requirements, the functional requirements for system must be clear indicated so that the user can know what the system all about. The following section will give detailed discussion about the function and non-functional requirements of this system.

3.4.1 Functional requirements

Functional requirement are what the user need for the system to work, they are requirements that we think of when describing systems function (Steve, 2007). Functional requirements are an expression of the various expected system behavior's expressed as services, tasks or functions which the system is required to perform. Functional requirement is what capture the intended behavior of the system (Ngwenya, 2010). The behavior is expressed in different ways; it can be a task, service, operation, etc.

The following functional requirements were determined in a brainstorming section with the stakeholders. The primary requirement and the main objective of this project is to integrate the legacy applications specifically e-Commerce and Moodle applications into SOA middleware called TeleWeaver. This is performed under ICT4D context, to help rural communities such as Dwesa. However, there are several functional requirements for this project; the following are the functional requirements for this project.

1. The system should provide communication between legacy applications and the TeleWeaver SOA middleware.
2. This system should provide vital communication between e-learning application, e-Commerce and the TeleWeaver middleware.
3. It should allow the art and craft entrepreneurs to upload their items to TeleWeaver middleware.
4. This system should enable TeleWeaver to transfer the uploaded item into e-Commerce application so that the end users can see all the recent uploaded products and send received message to TeleWeaver.
5. System also should be able to allow the teachers to upload information about various subjects into TeleWeaver then the middleware sends it into the e-learning application to update it with new information and the received response is also required.

6. The end users should be able to see the uploaded information from the legacy applications perspective.

7. The system should enable the easy way for administrator to update the system at any moments.

3.4.2 Non Functional Requirements

"A non-functional requirement is a statement of how a system must behave; it is a constraint upon the systems behavior" (Mark, 2009). Non-functional requirements are constraints of various attributes of the task or system. The non-functional requirement includes usability, reliability, interoperability, scalability and security. They are few non-functional requirement techniques that have been investigated in this project, which includes the following. These following are the non-functional requirements that were determined through the brainstorming sessions

1. Accessibility

In this project there are several ways of accessing system. Accessibility is the process of accessing the application without any barrier. The location of the system should be such that the administrator and art and craft people are able to access it easily without any constraints. There are several ways of accessing the system from the administrators' perspective. The end users can only access the system from the application point of view. The main advantage of integrating and modernizing legacy application is that the administrators have easy access to update and upload information to the system without any programming skills needed. It is also easy for a system to be maintained.

2. Scalability

The system should expand within the extension of development environment. Currently, the legacy application can only be maintained when the expert update it, it becomes difficult for new art and craft to upload product on their own, as much as they are computer literacy, due to complexities of the legacy applications it becomes difficult for the system to be updated. This system is expected to function in a way that developers find easy to maintain the application and modernized it.

3. Fairness

The major focus in this project is to provide a fair integration mechanism for SLL developers, whilst bearing in mind the economic capacities of the target population. Therefore, the system should consider programming background of the targeted area hence the integration method should be fair enough, so that it does not require more maintenance and to changing schemes that take into account the socio-economic status of the people around the Dwesa area. Due to complications associated with legacy application it makes business grow slow, therefore effectiveness of these applications in Dwesa community is so low.

4. Affordability

Considering the legacy application complications, legacy applications are very expensive when it comes to maintainability judging from the amount of budget that is put aside for maintaining legacy application. Legacy systems are not affordable, hence it is considered for SLL project to integrate this application so they can be sustainable and effective.

5. Flexibility

The system should be flexible enough to allow administrators such as rural ICT users to upload information and products on e-Commerce application through TeleWeaver Middleware. The system should have no impediments when administrators seek to upload product information and course details.

6. Compatibility

This system should be able to operate with other components that are design for interoperability, without having any barriers.

7. Extensibility

Having new components added to the system, the system should still function without drastic changes on the architecture. The new components can be added on the system without major changes on the system architecture.

8. Maintainability

This system designed should be easy to maintain. It should be easy to debug error and modify system or functions. The contextualization of this system is based on easy maintain of the applications. It should be easy to detect errors, fix errors and modify system without having constraints. The components in this system are independent, that result to better maintainability of the system.

9. Fault-tolerance

This system should be resistant and easy to recover from any components failure.

10. Modularity

The integration system comprises of well defined, independent components. All these components are loosely coupled. They are independent and that leads to easy maintenance of the whole system.

11. Usability

This system should be easy to use to administrators. This should have easy access to any components that user have a credential to access.

12. Effectiveness

Integration system is design to be easy to use, and cost-effective. This should lead to the effectiveness of this system. The integration of these legacy services should make this system easy and flexible. As the administrators will be able to upload their item and information, this will make the applications to be effective.

The importance of each of the above factors should reflect the objectives this system wants to achieve. Design consideration is a complex process, with user and server constraints to account for, complex business rules to accommodate, and the demands of the users to meet and there are certain operations that must remain the same in the system (Spool, 2007). Design consideration is a presentation of important information that needs to be considered when designing the system. This improves maintenance, flexibility and effectiveness of the system. It also reduces the chance of undesired outputs. For this system to design it, user interaction with the system has to be considered. As we mentioned on chapter three, that this system has to be affordable, hence we uses Free Open Source Software (FOSS). Due to the advantages of free open-source software, FOSS will be off advantage for this system. Also since this is meant for rural communities it has to be more flexible and easy to use.

This means that the software technologies used to develop the application should be those that are freely available and obtainable without any costly charges that could have negative effects on the revenues generated. This shows how the end user can access system without constraints. This also shows how this system accommodates service components on it.

For the integration application project to work, certain tools have to be considered. This enables the easy integration system. For this project there are several tools that make the system work, these tools will be discussed in the following paragraph.

3.5 Conclusion

This chapter presented requirement procedures and the various methods that have been used in this project. It also gave an explicit analysis of the whole system. The functional and non-functional requirements of the system have been discussed in this chapter. The way the users interacts with the system was also discussed with the diagrams that shows the user and administrator interaction with the system. The technology that was used during the development of the system was also presented in this chapter. This chapter demonstrated the mechanisms that were used to integrate legacy application under SLL. The next chapter will discuss the design of the system.

4. Chapter 4: System Design

The main focus of this chapter is to give explicit details on how this system looks like. The functionality of this system gives the conceptual framework of the system. The design of this project is represented in this chapter. The full details of how TeleWeaver connects with legacy applications are also discussed later in this chapter.

4.1 Introduction

This chapter explains how this project is designed. The main objective is to illustrate how legacy applications will be integrated into TeleWeaver Middleware under SLL intervention. It also addresses the issue of application design. The application design in this thesis addresses the issues of sustainability of legacy applications under SLL by providing a means of generating flexibility and sustainability through integration of legacy applications, and allowing administrators to upload and modifying existing services. The application design in this context is termed as integrating legacy applications.

There are certain design aspects that need to be considered for designing this system. This chapter discusses the system architecture and design frameworks and other aspects for designing the system on the following sections. The architecture of the system is shown later in this chapter.

4.2 System Architecture

System Architecture is a dynamic conceptual model that outlines the structure and behavior of the system (Francois 2010). System architecture is also an overall view of the whole system to ensure the robustness, cost-effective, flexibility, reliability and the strategic solution to the main problem. System architecture focuses more on the high level planning of the system, for development and implementation of the system, it elucidate the strategic steps for solution. It explains the overall steps for creation of solution from blueprints (Gilkey, 1960). System architecture also elaborates the different components and libraries organized to support reasoning about the structural properties of the system (Muller, 2010). The Figure 4.1 shows the conceptual design of the whole integration system.

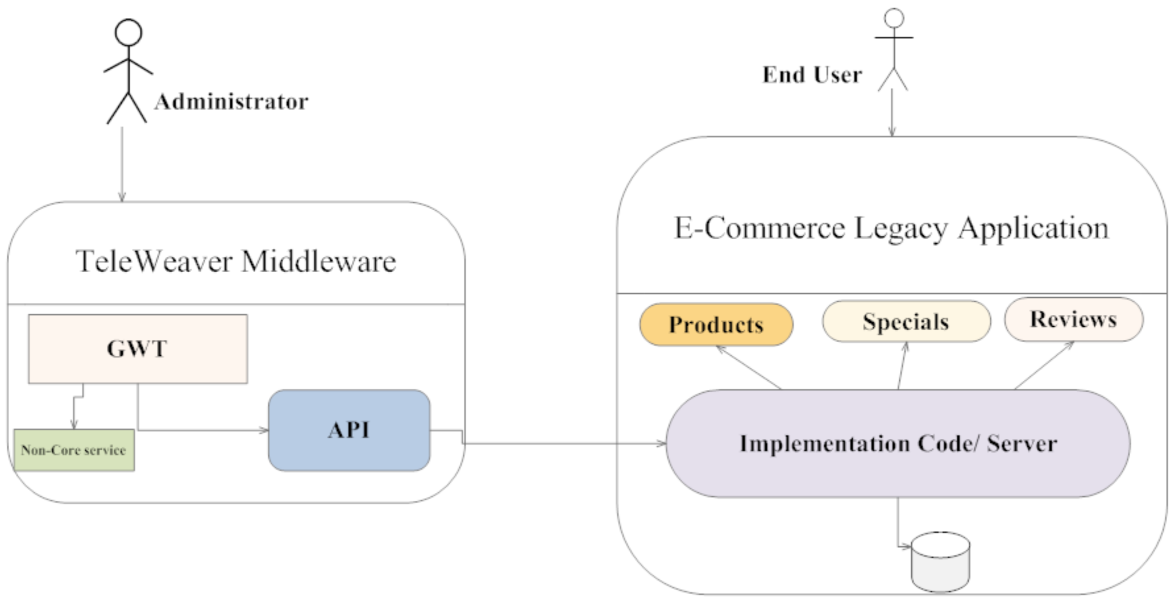


Figure 4.1 An e-Commerce System architecture

4.3 System Modules or Components

The system components presented in Figure 4.1 shows the main integration of legacy applications into TeleWeaver middleware. It illustrates the major interaction of the components within the system. This elaborates how the application integration is done, and how the components interact. It demonstrates how the third party applications are integrated into TeleWeaver middleware. The interconnection of Os-Commerce components, TeleWeaver, Services invoker, API are important aspect of this system. The following section illustrates how these components operate.

Google Web Toolkit is a view component that resides within TeleWeaver as stated in Chapter Two. This component allows the administrator to view services on TeleWeaver. It is used in this project due to its advantage of separating business logic from the logic layers. More details of GWT in chapter two of this research. As mentioned in Chapter Two API is another system component that is used in this research to connect TeleWeaver with third-party applications.

4.3.1 Third Party application Modules

The party modules are the modules that are available within the applications. These various modules contain different information about legacy applications, such as e-Commerce; contain information about product, products_id, reviews, specials and many more while Moodle contains information such as course_id, end-users' Id and course_infor. The modules they communicate with the API, as it is depicted in the Figure 4.1. The Figure 4.2 illustrates how the system was initially designed when using Nu-SOAP API.

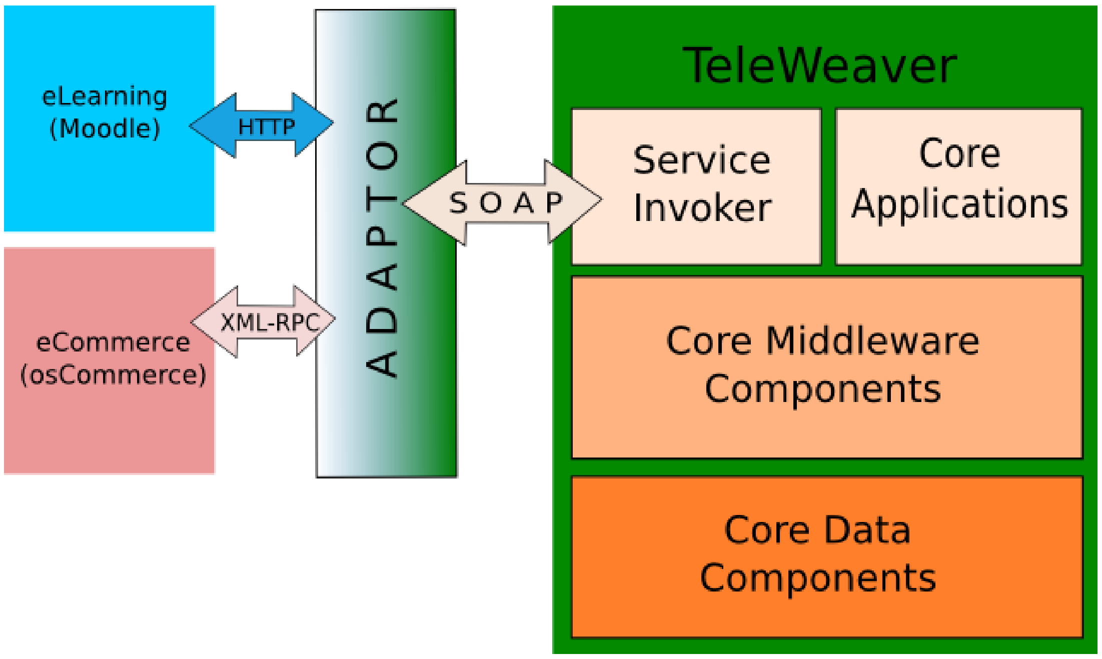


Figure 4. 2 Initial System integration using Nu-Soap

Figure 4.2 shows the conceptual idea of integrating these two specified legacy applications into TeleWeaver SOA middleware when using nu-soap API. The process of integrating legacy applications using Nu-SOAP API is flexible and sustainable, but however it mainly depends on how the middleware exposes its interfaces. A developer can smoothly integrate legacy applications using Nu-SOAP API**.**

4.4 System Integration

System integration is a process of integrating the whole system, to improve the functionality and effectiveness of the applications. System integration is defined as a process of gathering different component into one system and ensuring that the subsystems function together as a system (Francois, 2010). It is also known as a vital aspect of linking together various applications to act as a coordinated system. In this project the legacy applications are linked together with the TeleWeaver middleware to improve the effectiveness and flexibility of these applications. Since legacy services are difficult to maintain and unsustainable, integrating them into SOA middleware enhance its effectiveness.

This allows a client to view the desired data that stores in the applications database. This enables the effective functionalities of legacy applications and improves the performance of the applications. This leads to a better service delivery to the client means a flow of a business to art and craft entrepreneurs. However, for this system is sustainable by the developers, there is need for a design framework that present different layers of the system and their functionalities. The following section will explain more about design framework of this system.

4.5 Design Frameworks

"A framework is a set of common and prefabricated software building blocks that programmers can use, extend or customize for specific computing solutions"(Clifton, 2003). Frameworks are designed and built from collection of objects so both the design and code of the framework may be reused and be effective (Clifton, 2003). The framework of this system is designed using different layers, where developers can decide which layer is suitable for integration of the application. The design of the TeleWeaver architecture is meant to have three different layers, presentation layer, business logic layer and the third part service. The Figure 4.3 illustrates the different types of layers that available in the Middleware, where integration of application will take place. These three layers are discussed in the following section.

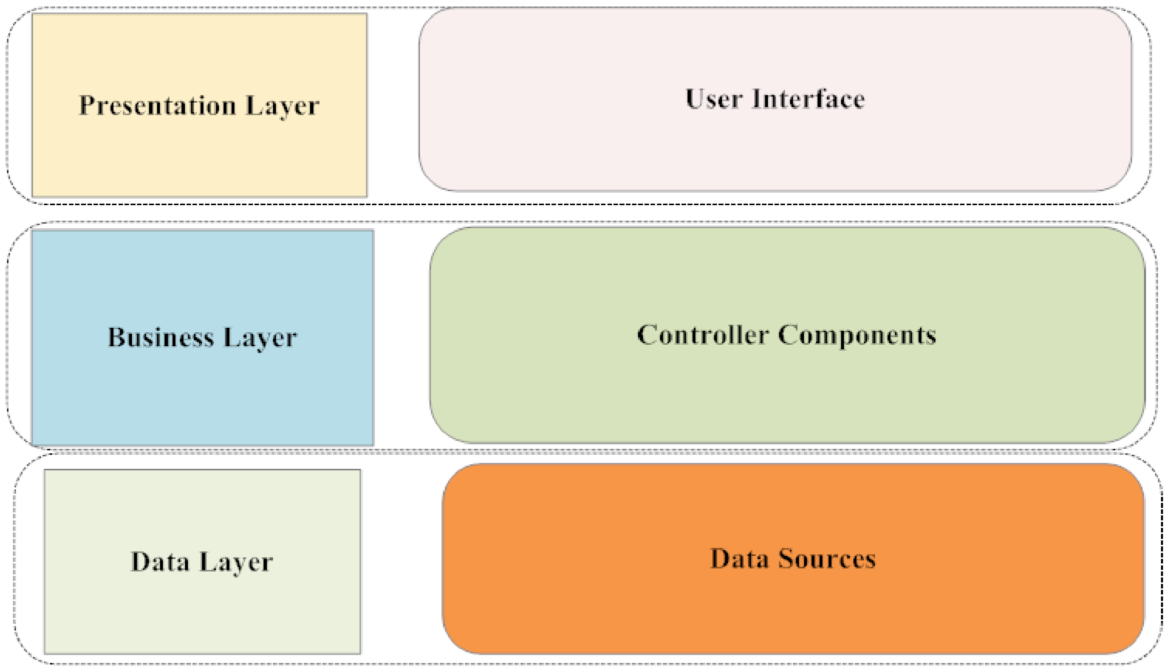


Figure 4.3 System design frameworks

4.5.1 Presentation layer

This presentation layer is the top layer in this system where users can view the services that are available in the middleware. This is the layer where integration of application takes place. This is where an administrator checks all the services that middleware offers and select the service that is of interest to him/her. In this layer the administrator is able to upload product and information that seek to appear on the application. It is this layer where the administrator maintains the services, especial the legacy services. These components add more flexibility to the system, as it displays the data to the administrator and fetch a data from the application. This layer enables art and craft and school teachers to delete unwanted data on the application, also enable them to add data to the application without hindering. Although the main functionality of this component is to display service to the user, it also allows art and craft to add more items to the e-commerce application. This layer is separated from the main core service component. It only displays the data and the major functions are operated by the business logic layer.

4.5.2 The Business logic

The Business logic is the layer where the main core components take place. This is where the API is residing on. It is where the main work occurs. It is directly connected to the third party applications. This core services component controls the application's functionality by performing detailed processing of business rules and task-specific behavior. It handles and processes user requests and gives the output back to the user through appropriate views. This is done through appropriate models and associated views. In our architecture this component is mainly responsible for transporting data from the GWT presentation layer to the database of the third part application. The business logic consists of different call methods such put, delete, get and post method. These methods are used for creating, updating and getting data and view data from the database of the application.

The business layer is the layer that directly interacts with the third part service. From the API that residing on the core service component which is a business layer, it then communicates with the implementation code of the third part application using calls methods. The implementation part of the third party application, it then responds to the API and return the requested data. However, when the administrator wants to upload data, she/he uploads on the presentation layer of the middleware where GWT resides. The GWT send information directly to Business logic layer where the API is sitting on, from the business layer then the link to the legacy service implementation code. In this architecture the database of the middleware does not perform the major role because from the business logic layer the communication goes straight to the legacy application that is integrated.

4.5.3 Third Party Application

The third party application is the application that is stand-alone application. It is the application that we seek to integrate it to the middleware. In this research the third- party applications that we focus on are two disparate application i.e. e-Learning and e-Commerce. These applications are the applications that play a fundamental role in the development of the rural areas. The e-Learning application is a service that provides learning tool to the students, in that way it enhances the education standard of the rural areas. The teachers are able to give more information to the student using this mechanism. The unavailability of the teacher will not have great impact on the student as they are able to access information from the Moodle application. Moodle is e-Learning application has been developed for the Dwesa community. This enhances the education standard of the rural communities

The e-Commerce application is a legacy application that enables rural entrepreneurs to sell their products online. As mentioned in Chapter Two that Dwesa is a community that has many art and craft people. These people design colorful items that bring attention to the tourist. SLL has developed an e-commerce application for these people so that the business can flow. Even for people who are outside Dwesa can be able to buy product online. However that was a vital plan of developing these community, but since these application are legacy applications are hard to maintain and unsustainable. Since people in Dwesa do not have good programming back ground and computer background, it is difficult for them to keep the system up to date. As these applications are stand-alone application, it is assumed that for these applications to be effective and sustainable they need to be integrated into TeleWeaver SOA middleware.

These applications are integrated using API libraries, to allow communication between different components and TeleWeaver middleware. The API as mentioned in previous section it allow communication between TeleWeaver and the third party applications components. The Figure 4.3 shows how this application integration works with third part application. It illustrates more on how these legacy applications are integrated to TeleWeaver SOA middleware.

4.6 User Interface

The system's user interface must allow users to interact or use it without major hurdles, as a result, a very intuitive but easy to use GWT. Interface must be employed to enable users of the system to operate the application. The GWT interfaces assist developers to speed up the development as it separate patterns. This interface eliminates the bad design patter, by separating presentation logic from the programming to the interface. This helps to maintain look for the application. It minimizes the errors that can be created when fixing the interface. As it mention on the above section, having interface separated from the core programming code, accelerate the developers work. Also it presents the neat and stylish interface. This interface does not know anything about the implementation code of the application, hence it is easy to maintain this system.

The system consists of two kinds of users, administrators and clients with different levels of privileges. Administrators have all privileges in the system which ranges from accessing system configuration interfaces to assigning profiles to clients. In this case, the administrators include teachers in the schools and the developer of the system. Clients are end users who consume the services provided by the system. As a result they have limited levels of system privileges to prevent the unauthorized use of the system, including accessing other people's profiles which might lead to a violation of privacy rights.

4.7 Use-case Scenarios

A use case is a description of the interactions and responsibilities of a system, the system and actors (Daryl, 2004). An actor may be a person, a group of people that uses the system. Use case scenarios are defined as the behavior of the system as it responds to a request from outside (Sparx System, 2007). The use-case scenarios are used to identify system components and processes and can further help to identify system requirements and functionalities when put to use (Pade and Palmer, 2009). Use cases also describe the responsibilities of the system under design, without getting into implementation techniques. The use case describes the various sets of interactions that can occur between the various external agents, or actors (Alistair, 2000). It is known as a list of steps for interaction between the user and the system.

In this project the use-case scenarios are based on the roles of administrators and clients as the primary actors. Figures 4.4 and 4.5 demonstrate the function of administrator and user's on applications, e-commerce and Moodle application.

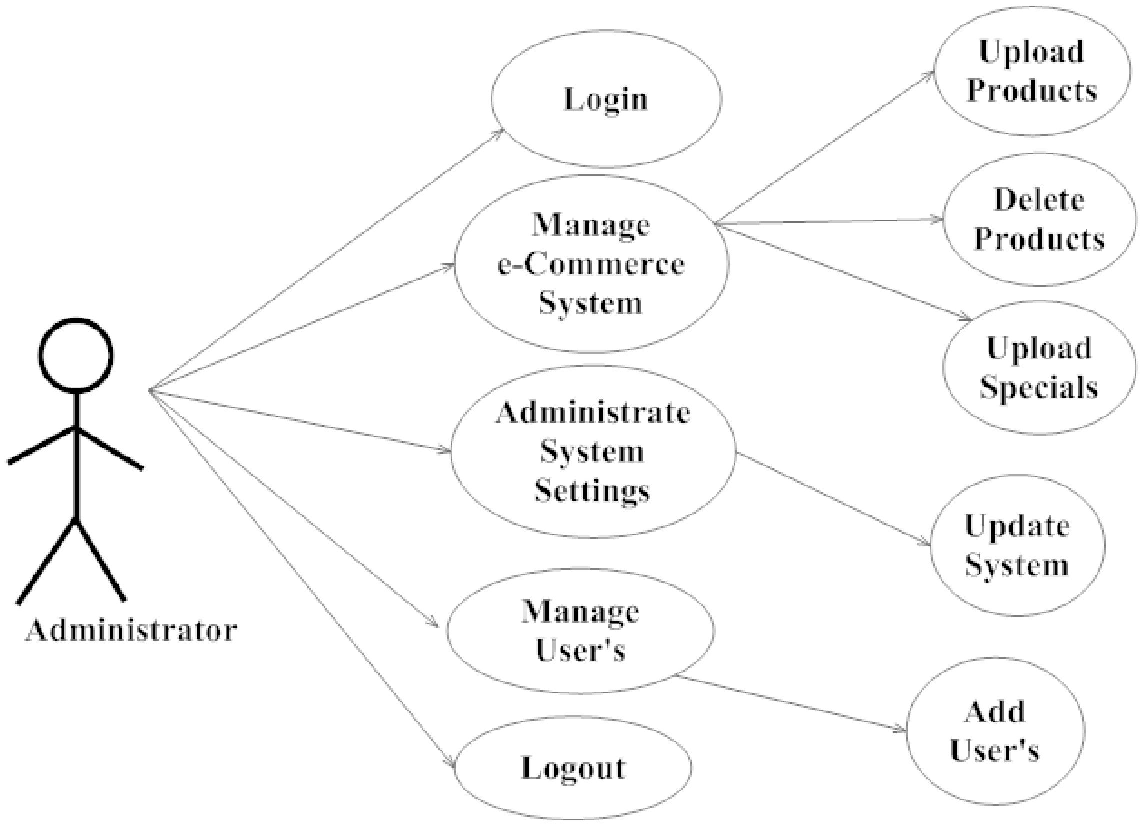


Figure 4.4 Use case Diagrams for e-Commerce application

The following use case diagram illustrates the way the user interacts with Moodle application and gives details of Moodle application.

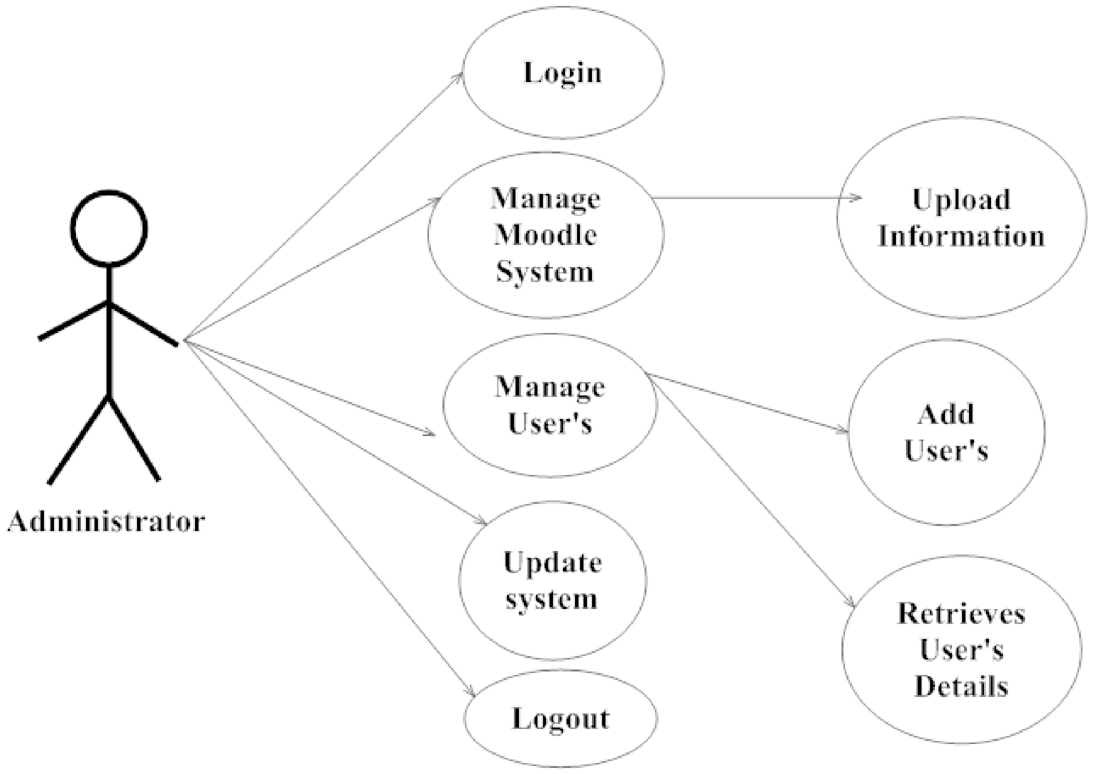


Figure 4.5 Use Case Diagram for e-Learning Moodle application

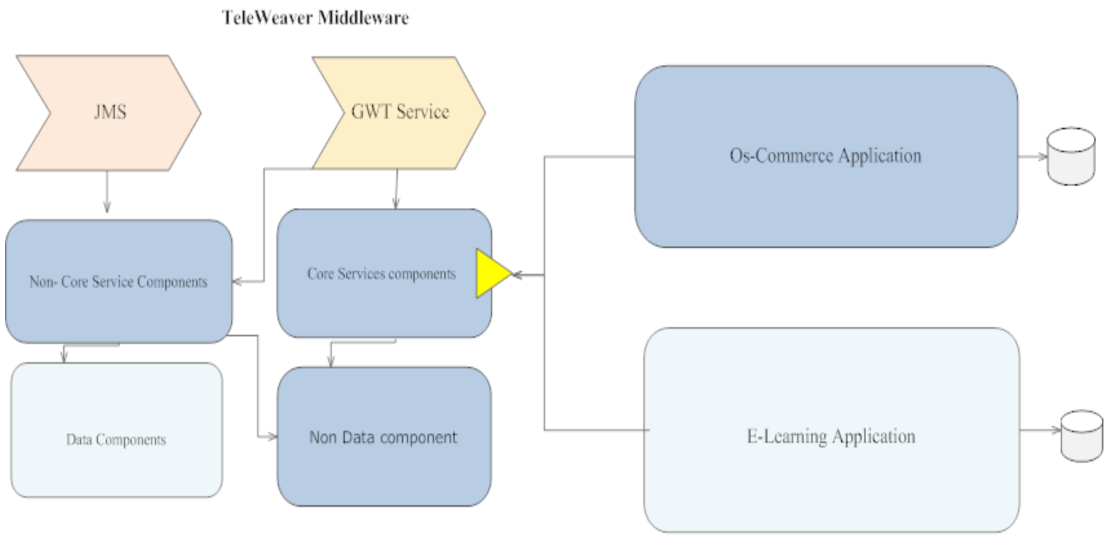


Figure 4.6 Application integration system

Figure 4.6 shows how the integration takes place and how these two applications are communicating with TeleWeaver middleware. This figure shows the link between TeleWeaver and Legacy applications.

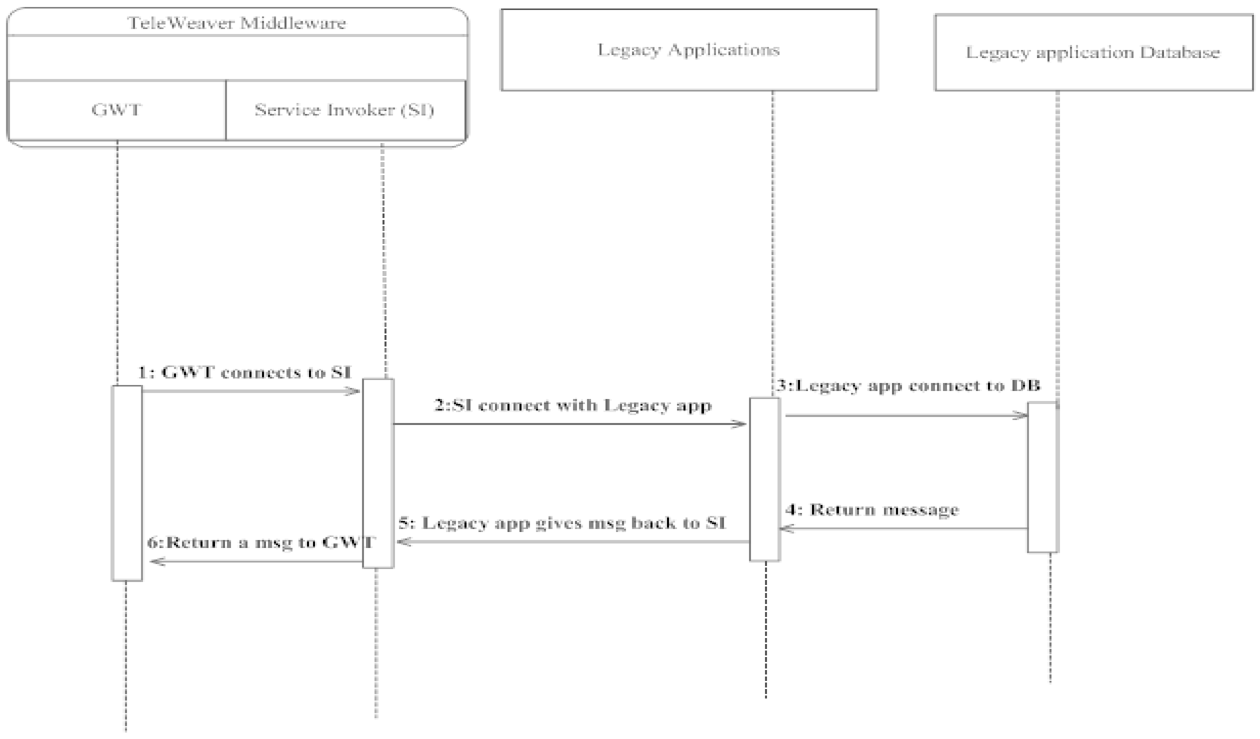


Figure 4.7 Sequence diagram for application integration system

The Figure 4.7 shows how the entire system operates it explains the process of communication between an administrator, TeleWeaver Legacy application, service invoker and REST API. This sequence diagram explains in detail how are the operations linked and how is the communication take place.

This sequence diagram illustrates the link between TeleWeaver and Moodle legacy application. It describes the process of communication and the legacy application links with TeleWeaver. This diagram shows how the Moodle application interacts with the administrator via TeleWeaver middleware.

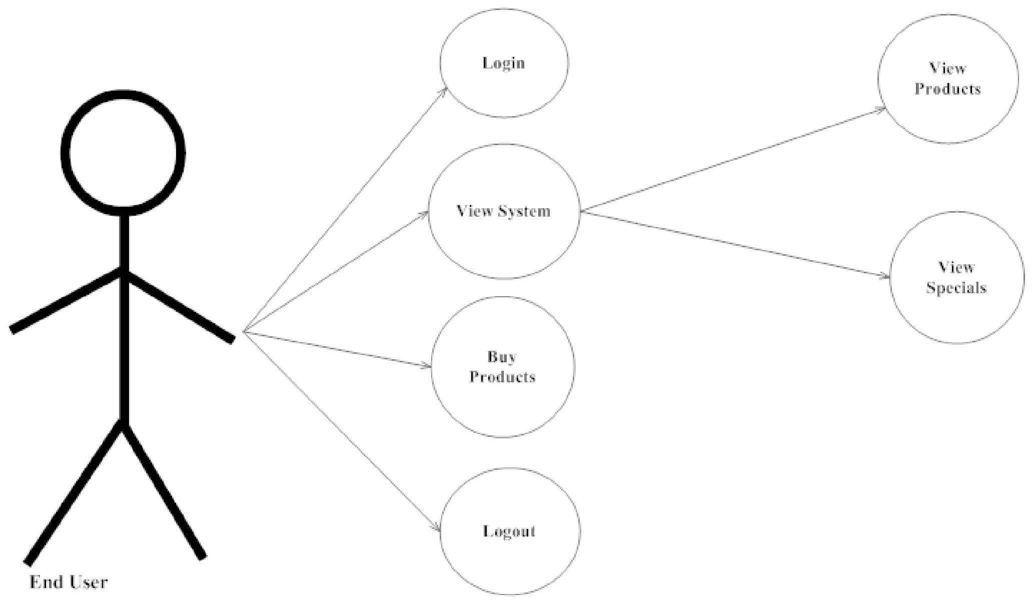


Figure 4.8 End User diagram for e-Commerce applications

The Figure 4.8 illustrates how the end-user interacts with e-Commerce applications. The use case diagram show how the communication goes and it gives a detailed discussion of interaction between the end user and the system.

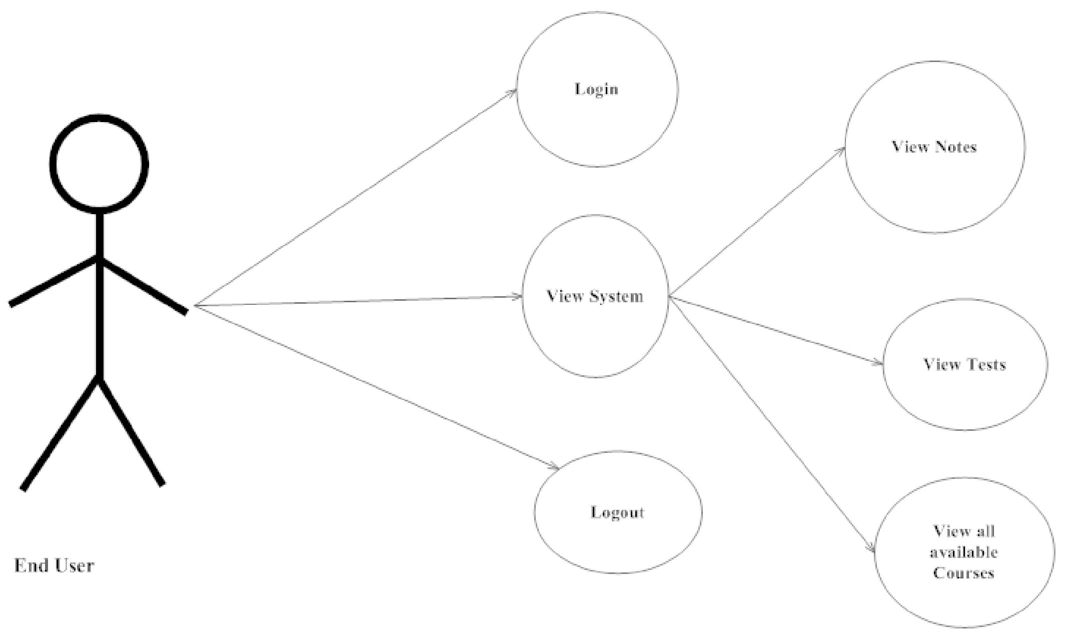


Figure 4.9 End –User diagram for Moodle application

The above diagram shows how the end users interact with Moodle application. The use case diagram gives details on how the uses interact with Moodle system

4.8 Conclusion

This chapter has presented the system design which gives an overview of the system architecture. The design considerations and the goals associated were explained. Various system components were explained and the way which these legacy applications are integrated. The emphasis was on how these legacy applications are integrated on TeleWeaver middleware and how these components communicate. The method of integrating these services was also stated in this chapter. The vital way of integrating the stand-alone application into SOA middleware was discovered and discussed in this chapter. The next chapter explains the implementation of the system.

5. Chapter 5: Implementation and Development of the system

The previous chapter explained the design of the system. This chapter summaries the implementation and development process of integration of legacy applications into the SOA middleware. The details of connecting middleware and the legacy applications are explained explicitly. The connection measures adopted in the system are also explained. Finally, the usage of the system by administrators, teachers and the art and craft members is demonstrated.

5.1 Introduction

This chapter presents the implementation and the development of this system which was executed by two dynamic phases. The first phase involves the basic installation of software packages and client-server architecture. This includes installation of Apache as a web server, PHP as a server side scripting language and MySQL as a database management system. There is PhpMyAdmin which provides Graphic Users Interfaces (GUI) for administration of database. However, this is followed by installation of legacy applications which include Os-Commerce and e-Learning applications. This is accompanied by installation of REST API which is called Slim. Slim is a REST API that resides on top of Os-Commerce and e-Learning application which uses REST client Firefox adds on to consume all the components that available on these applications.

The second phase is where the deployment environment, which is TeleWeaver Middleware have to be installed and development environment has to be created. This phase involves the connection of legacy applications with TeleWeaver middleware. This includes connection of legacy applications with REST API which seeks to connect legacy applications with TeleWeaver SOA middleware. This involves the construction of Google Web-tool Kit (GWT) which allows developers to upload items on it. GWT is the GUI on TeleWeaver side where a developer can upload information so it can go through Slim API to legacy application. This interface allows the developer to retrieve information from legacy services. This horizontal dimension is where SOA middleware links with the dynamics of the legacy applications and with all the assets located on the development and deployment environment, to connect the middleware with the legacy services.

5.2 Legacy applications

It must be noted that these two legacy application use PHP scripting language which make it easier for Slim API to connect with every components of legacy applications. These applications constitute of various components that contains different functionalities. The two applications stated in the above section consist of core service components that connect to each legacy applications database. However, these components are arranged in different ways. For these applications to connect with the middleware, certain adjustments have to be made. A service that leverages these legacy application components is essential. The following section discusses more about the arrangements of the core services inside legacy application. These applications contains of SQL database which comprises of many tables that hold different function control parameters. Section 5.3 discusses more about system organization of legacy applications and connection.

5.3 System Organization

This section is about piloting a conceptual framework and generating pertinent information that will inform best integration of legacy application. Fundamentally, this project is about assisting the Dwesa communities to develop their social capital as a foundation on which to build-up and use community assets in a sustainable manner on which they can run their business, also to enhance the education system. Therefore, integrating these legacy applications involves innovation of these existing services and leverage's the existing skills. A brief presentation on the organization of the components inside existing applications and the arrangement of platform that allow communication between components and the middleware is highlighted in this section. The stated existing applications were design by certain language, these services has dynamic components that connects to SQL database. However, these components were arranged in a certain path. The next section covers on the application arrangement, indicating where the Slim API resides on the legacy applications.

5.4 Moodle Application

Moodle is a legacy application that is developed using PHP scripting language (Dalvit el at, 2007). This application consists of different core service components that have different functionalities. As mentioned in Chapter Three, this service is developed to assist in improving the education system of impoverished communities. However, this application needs to be integrated into TeleWeaver in order for it, to be effective, extensible and sustainable. The following section represents the arrangement of Moodle components. This service needs service adjustment for it to connect with the middleware. An API has to be developed to allow smooth integration with TeleWeaver. The following snap shoot shows the arrangement of components inside Moodle. These components can be retrieved by the developer on the TeleWeaver side, once the application is integrated. The developer sitting on TeleWeaver has to update information on Moodle so the end users can retrieve the new updated information.

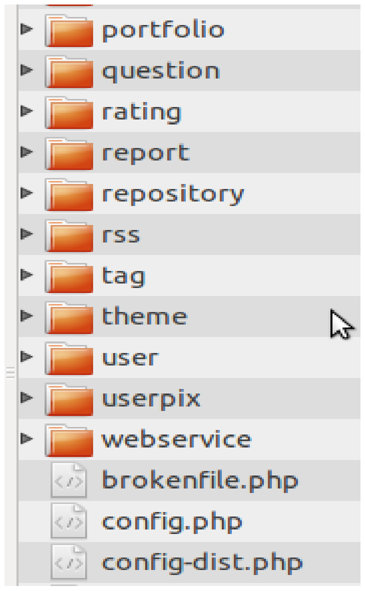
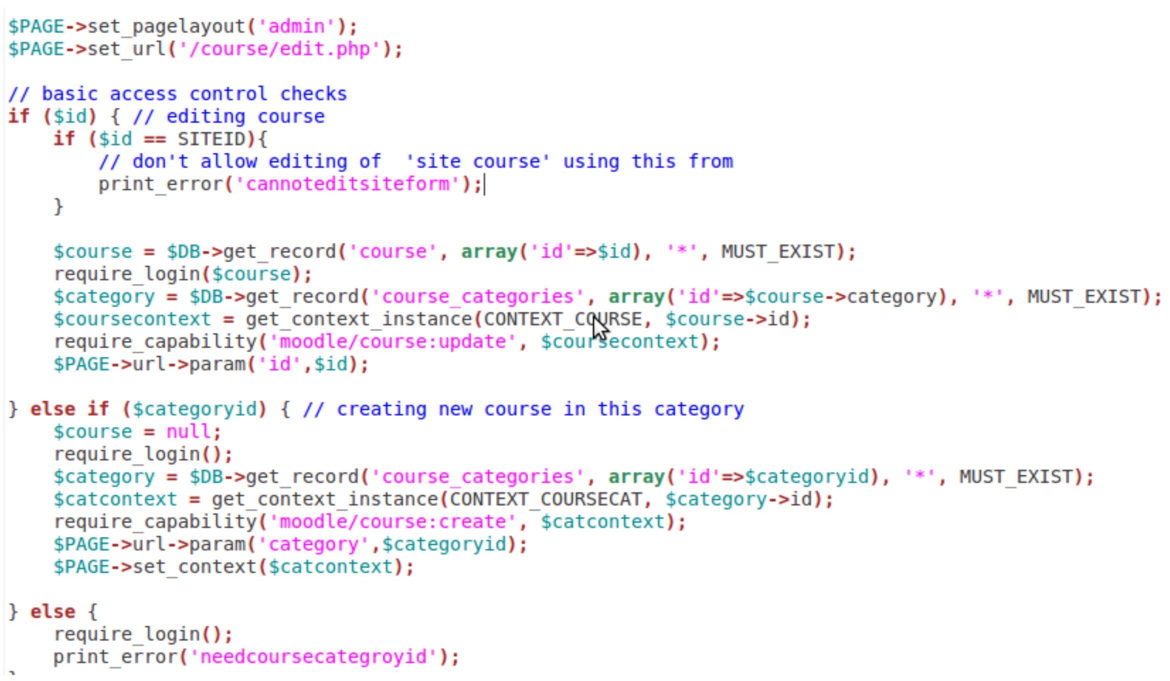


Figure 5.1 Core components of Moodle Application



```php
$PAGE->set_pagelayout('admin');
$PAGE->set_url('/course/edit.php');

// basic access control checks
if ($id) { // editing course
    if ($id == SITEID){
        // don't allow editing of  'site course' using this from
        print_error('cannoteditsiteform');
    }

    $course = $DB->get_record('course', array('id'=>$id), '*', MUST_EXIST);
    require_login($course);
    $category = $DB->get_record('course_categories', array('id'=>$course->category), '*', MUST_EXIST);
    $coursecontext = get_context_instance(CONTEXT_COURSE, $course->id);
    require_capability('moodle/course:update', $coursecontext);
    $PAGE->url->param('id',$id);

} else if ($categoryid) { // creating new course in this category
    $course = null;
    require_login();
    $category = $DB->get_record('course_categories', array('id'=>$categoryid), '*', MUST_EXIST);
    $catcontext = get_context_instance(CONTEXT_COURSECAT, $category->id);
    require_capability('moodle/course:create', $catcontext);
    $PAGE->url->param('category',$categoryid);
    $PAGE->set_context($catcontext);

} else {
    require_login();
    print_error('needcoursecategroyid');
```

Listing 5.1 Moodle core service component back end connection with database

Listing 5.1 shows the back end of the code that creates and edits the course in the Moodle application. It connects the core service components of Moodle into my SQL database. This elaborate the dynamic transition that takes place for this integration system, the core services of existing services communicate with database, then the RESTful service, connects with these components using API. The SOA middleware make use of the method call, to communicate with API, therefore the API itself send a message to the required service using URI of the service required. This makes the integration system effective, reliable and extensible. However, using restful services makes integration process sustainable. More illustration of the RESTful services and the connection of SOA middleware will be explain on the following sections.

This section shows the snap shot view of the system organization using the recommended technologies discussed in Chapter Three. As mentioned in Chapter Three, slim API is utilized to connect legacy application components so when the rest client seeks to consume information can be able to retrieve it without any hindering.

5.5 Os-Commerce Legacy Application

The Os-Commerce application is the application that uses a SQL database to store information about its product. It is developed using a PHP script language. It consists of multiple tables in a database. These dynamic tables contain different information for different components. However, for the middleware to connect to this legacy application there is certain criteria that need to be followed. Os-Commerce consists of various components, for different functionalities. Each service has its own file, which contains information about the services. For e-Commerce application to connect with TeleWeaver middleware, REST API have to be developed which uses slim to connect with different components of legacy applications. However, Slim API resides on top of e-Commerce application to allow smooth connection between different components. Figure 5.2 elaborates how Slim API resides on Os-Commerce application and the structure of e-commerce files.

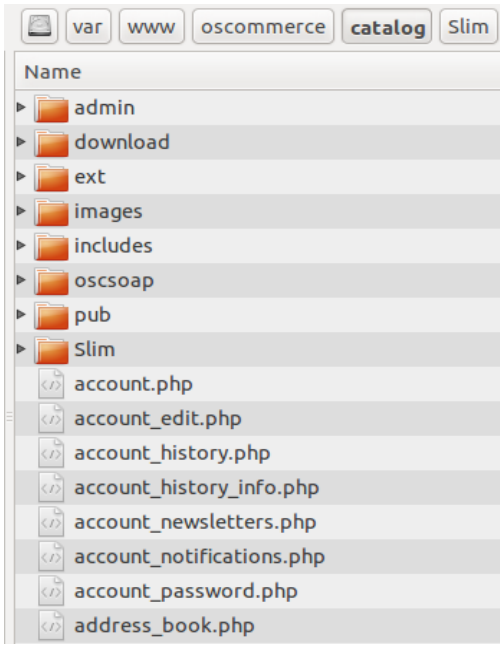


Figure 5.2 Structure of the folder system on Os-Commerce

5.6 Developing Application using Slim API

Slim is a PHP micro- framework that is designed to create simple and flexible applications and it comes with the minimal components, such as response, request, view and route. To get started with Slim, PHP 5.2 + is required which allows developers to write an applications in a procedural style or in 5.2+ style.

In Slim framework the general sequence of developing an API that connects with core components of legacy applications is as follows.

- Download Slim package.
- Install the package.
- Configure / Set-up your project.
- Start developing an application.

The above steps are the essential criteria that need to be followed when a developer seeks to use Slim package for connecting dynamic applications. On the demonstration below, the developer needs to remove all the content that is contained in the index.php and begin to write his/her own code. The file index.php act as a bootstrapping, hence the developer needs to set-up his/her own route when want to use Slim. Figure 5.3 is the demonstration of how slim package is structured and the components contain within Slim package.

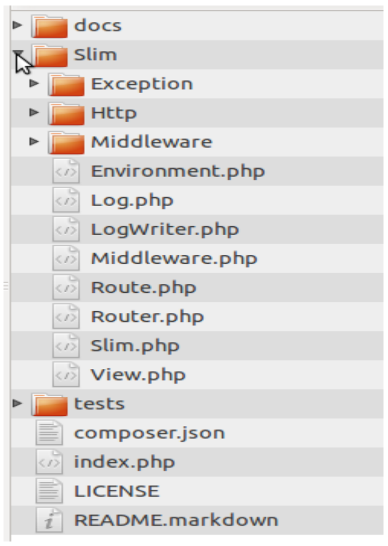
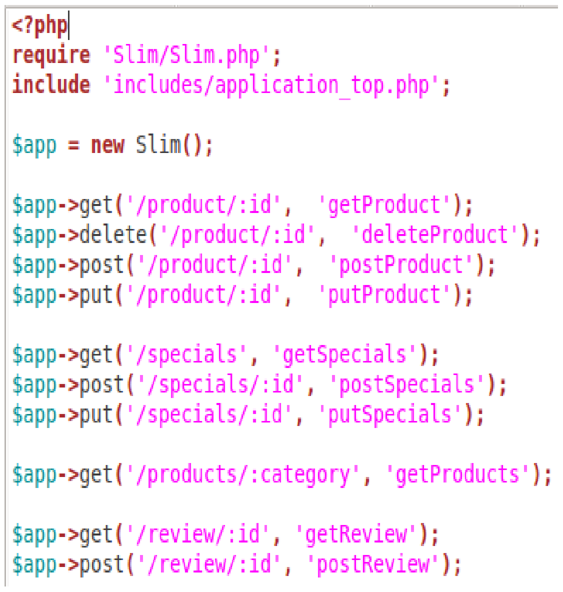

```
docs
Slim
  Exception
  Http
  Middleware
  Environment.php
  Log.php
  LogWriter.php
  Middleware.php
  Route.php
  Router.php
  Slim.php
  View.php
tests
composer.json
index.php
LICENSE
README.markdown
```

Figure 5.3 A Structure of Slim Packages

5.7 Developing a Slim API

In RESTful web services the main arena that is required when developing a Slim API is to include all the vital libraries that will make an application function properly. This includes the perfect routing of the files, the correct methods that are used when want to consume the service component of legacy application such as post, put, get and delete methods. However, the snap shot below demonstrates how to use methods on slim API. This presents the methods that are used in the e-Commerce application. This also shows how to register the methods of slim API.



```
<?php
require 'Slim/Slim.php';
include 'includes/application_top.php';

$app = new Slim();

$app->get('/product/:id',  'getProduct');
$app->delete('/product/:id',  'deleteProduct');
$app->post('/product/:id',  'postProduct');
$app->put('/product/:id',  'putProduct');

$app->get('/specials', 'getSpecials');
$app->post('/specials/:id', 'postSpecials');
$app->put('/specials/:id', 'putSpecials');

$app->get('/products/:category', 'getProducts');

$app->get('/review/:id', 'getReview');
$app->post('/review/:id', 'postReview');
```

Listing 5.2 Slim API for registering methods

5.8 API that Connects with Legacy Application

Slim API has several dimensions of connecting with existing service components, the initial criteria is to register the services and declare the method that will be used. The second part is to get a function that connects with the database of the component that needs to be retrieved. For the product to be displayed the connection with the database is essential. The snap shot below elaborates on how to connect slim API with existing service database, when using Get method.

```
function getProducts_new(){
$languages_id = 1;
$products_new_array = array();

  $products_new_query_raw = "select p.products_id, pd.products_name, p.products_image, p.products_price,
m.manufacturers_name from " . TABLE_PRODUCTS . " p left join " . TABLE_MANUFACTURERS . " m on (p.manufac
TABLE_PRODUCTS_DESCRIPTION . " pd where p.products_status = '1' and p.products_id = pd.products_id and p
p.products_date_added DESC, pd.products_name";

$products_new_query = tep_db_query($products_new_query_raw);

$allproducts_new = "<br />";

    while ($products_new = tep_db_fetch_array($products_new_query)) {
        $allproducts_new = $allproducts_new . $products_new['products_name'] . " <br />";

}

   echo json_encode($allproducts_new);
}
```

Listing 5.3 SLIM API connection with e-Commerce Database

5.9 Rest Client

This section presents the snap shot view of rest client add on, which is responsible for consuming the components that are registered or declared on slim API. REST Client Firefox add on, require a correct URI in order for it to consume the services. The below snap shot retrieve the information of the product, and allow the upload of the items depending on the method assigned on the API. Below is the snap shot that presents the rest client when it is retrieving the information about product that is on e-Commerce application.

The snap shot below shows the rest client with request methods and the URL. This elaborates how REST client functions.

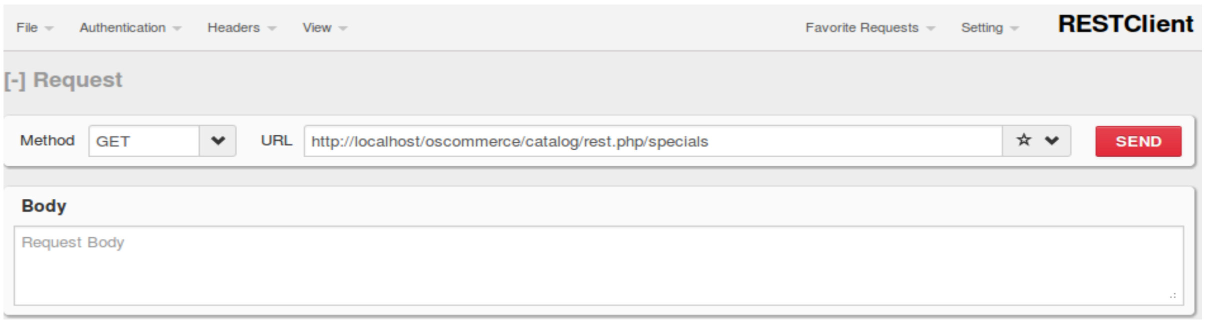
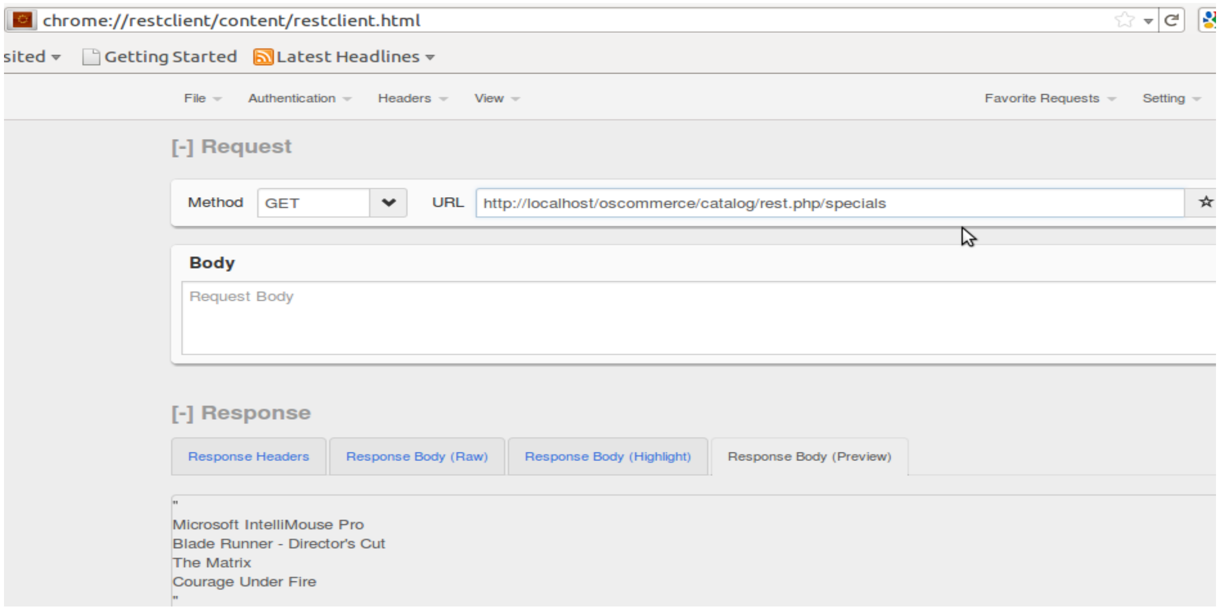


Figure 5.4 Basic Rest Client structure

5.10 Rest Client method for Consuming Services

It must be noted that rest client is the main method that allows the administrator to consume all the services that are available on legacy application. It is the one that allows an administrator to post new products on the catalog. This method allows the administrator to view all the products that are available, to update the system, to delete unwanted products, and to view the details of all customers.

The screen shot below presents the rest client method for retrieving the specials that are available on e-Commerce catalog.



Method GET URL http://localhost/oscommerce/catalog/rest.php/specials

```
"
Microsoft IntelliMouse Pro
Blade Runner - Director's Cut
The Matrix
Courage Under Fire
"
```

Figure 5.5 Rest Client that retrieves the list of specials in the catalog

5.11 The e-Commerce Client Side

Figure 5.6 shows the e-Commerce interface on the client side. This is how the integration process appears on the client side after all. The client can only see this side of the application; the connection transaction is hidden from the client.

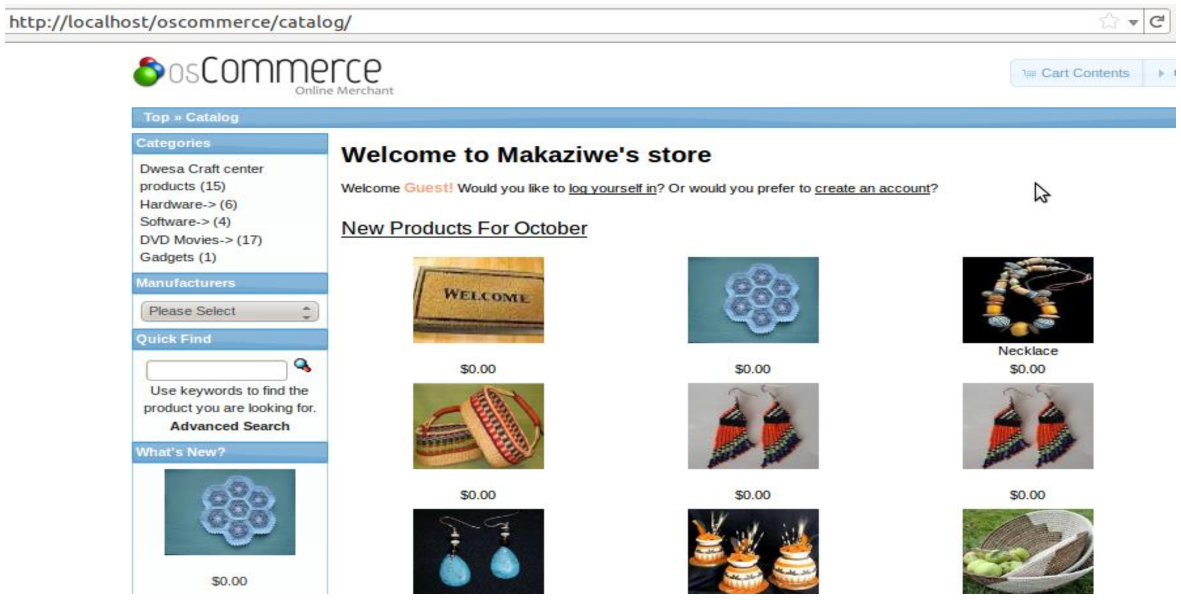

http://localhost/oscommerce/catalog/
osCommerce Online Merchant
Cart Contents
Top » Catalog
Categories
Dwesa Craft center products (15)
Hardware-> (6)
Software-> (4)
DVD Movies-> (17)
Gadgets (1)
Manufacturers
Please Select
Quick Find
Use keywords to find the product you are looking for.
Advanced Search
What's New?
$0.00
Welcome to Makaziwe's store
Welcome Guest! Would you like to log yourself in? Or would you prefer to create an account?
New Products For October
$0.00
$0.00
Necklace
$0.00
$0.00
$0.00
$0.00

Figure 5.6 The e-Commerce Application

All the products are uploaded on TeleWeaver using GWT, and go through Slim API to e-Commerce application and the end user's view the application.

5.12 GWT

Google Web Tool kit is the graphic user interface that separates the presentation logic from the business logic. It assists developers to design a clean and simple application. This helps developers to speed up the development process. In this system GWT is used by administrators and rural ICT users to upload information and images for these legacy application. The administrator uploads item on GWT. As this system is designed to assist rural community members, the interface must be designed in a manner that will be simple and flexible for rural community member.

5.13 TeleWeaver connection with 3rd part applications

TeleWeaver is an open source technology that uses a standard compliant and tested packages, it is an SOA middleware that developed using Java programming language. TeleWeaver uses OSGI as a container. This middleware offers a blueprint for services to be integrated to it. It uses certain software packages and programming language such as JAVA EE, Spring-Framework, and Apache-CXF.

However, this middleware consists of several core service components. It comprises of core service components that allows developer to view the services, it have non-core service components, extension data component and core data components. The following screen shot demonstrates the architecture of TeleWeaver middleware. Using this middleware the integration took place at the presentation layer of the middleware, where GWT use a method call and rest API to connect to legacy applications.

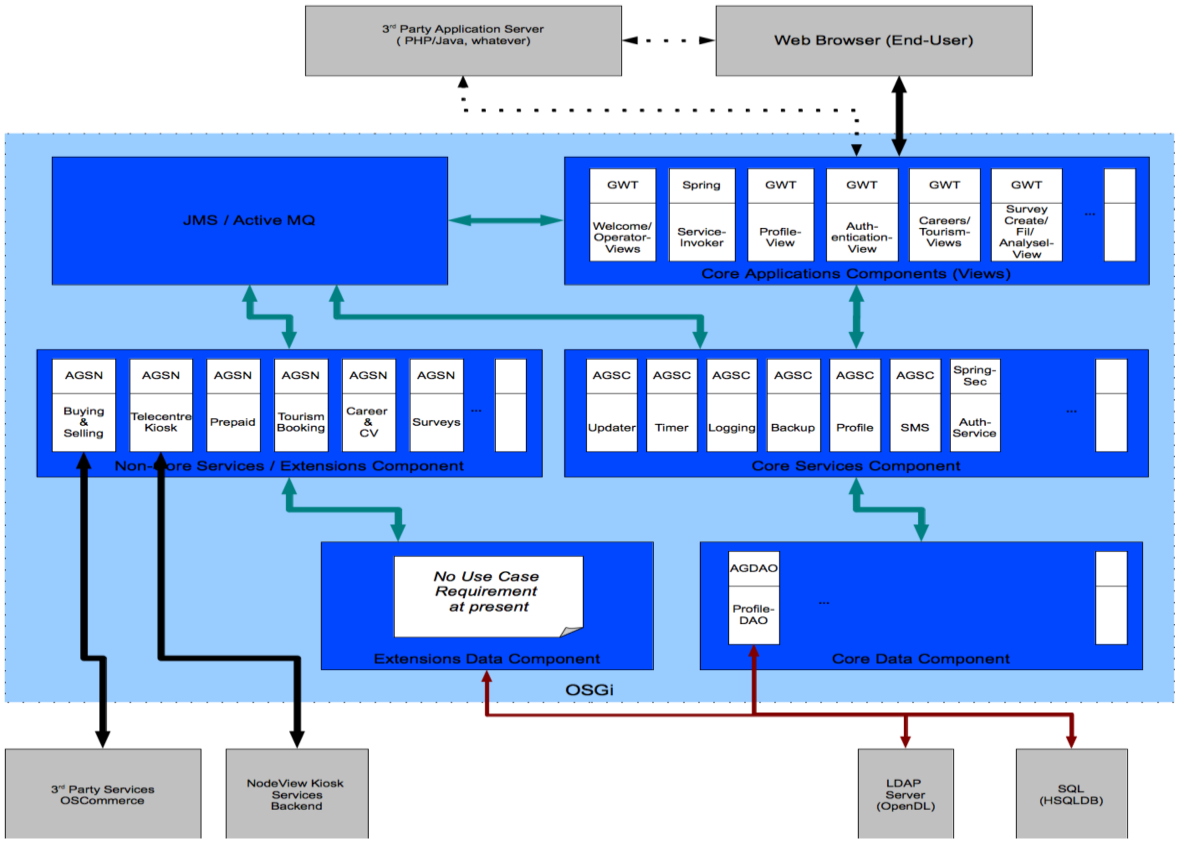
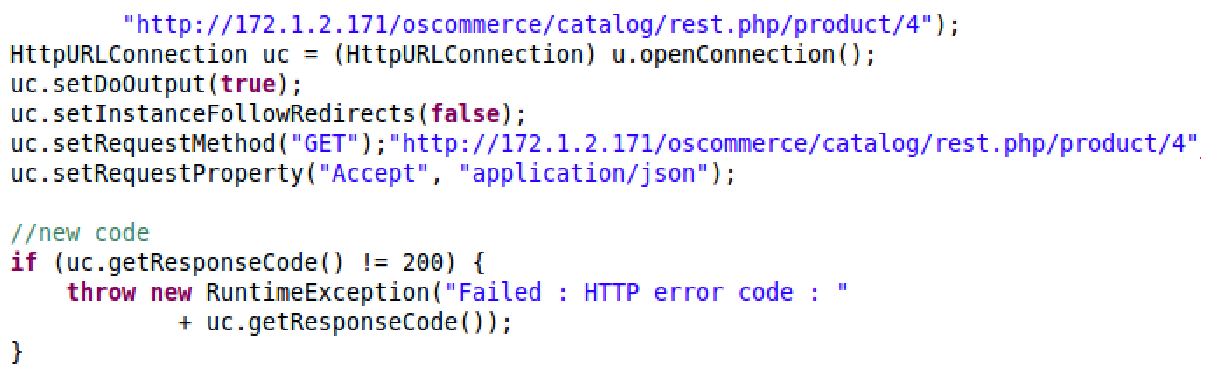


Figure 5.7 TeleWeaver Architecture (Reed House Systems TeleWeaver Middleware v5.3. R. Werteln)

The Slim API connects with TeleWeaver using a method call. Figure 5.7 shows how TeleWeaver connects with the third- party applications. As stated in Chapter Four that a certain technology needs to be used, is restful web service to connect with legacy applications. The data stored in each legacy database is accessed by core service component of each application. Therefore slim API links with components of each legacy service. The most important protocol in slim API is route and the URI of each component. TeleWeaver connects with the each service using URI. The code below that elucidates all the details of the TeleWeaver connection with legacy applications.



```
        "http://172.1.2.171/oscommerce/catalog/rest.php/product/4");
HttpURLConnection uc = (HttpURLConnection) u.openConnection();
uc.setDoOutput(true);
uc.setInstanceFollowRedirects(false);
uc.setRequestMethod("GET");"http://172.1.2.171/oscommerce/catalog/rest.php/product/4"
uc.setRequestProperty("Accept", "application/json");

//new code
if (uc.getResponseCode() != 200) {
    throw new RuntimeException("Failed : HTTP error code : "
            + uc.getResponseCode());
}
```

Listing 5.4 TeleWeaver connection with e-Commerce legacy application

Listing 5.4 demonstrates how TeleWeaver connects with legacy application. This method has to be implemented for smooth communication with slim API in third- party application. The method of rest must be clearly stated, if developer seeks to post information to the application, he/she must state the method on setRequestMethod. The code above elaborate the essential method that one can use when want to retrieve information from the application and the specific the URI of the service that one seeks to retrieve. The last method that is off value when using this Slim API is to acquaint the setRequestProperty on the above code, which data type is being used, is it Json or XML.

5.14 TeleWeaver Components

TeleWeaver is installed using and configured. It uses bundles and equinox to communicate with the services inside OSGI container. It consists of numerous services that are used when one want to run his/her bundles. There is a drop-box where the developer drops the jar file of each project that has been developed. However, after dropping jar file on drop-box, there is a configuration file that keeps records of the logs. Each time the jar file is dropped in a drop-box the logs are created on the configuration file. However, before the developer run then service on browser the logs must be deleted, then restart TeleWeaver, thereafter can run the service of the browser. The screen shot shows the arrangement of TeleWeaver components.

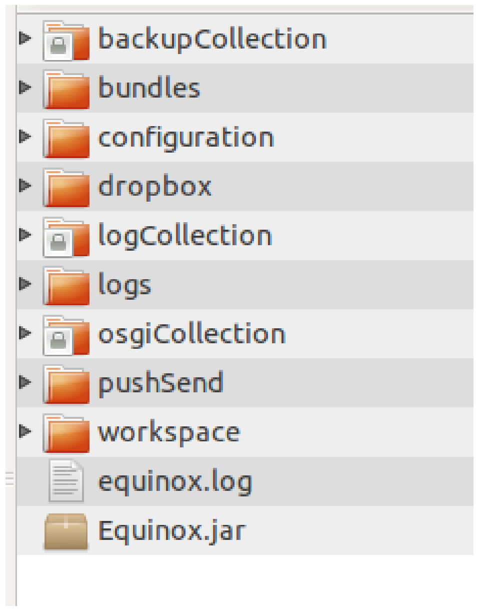
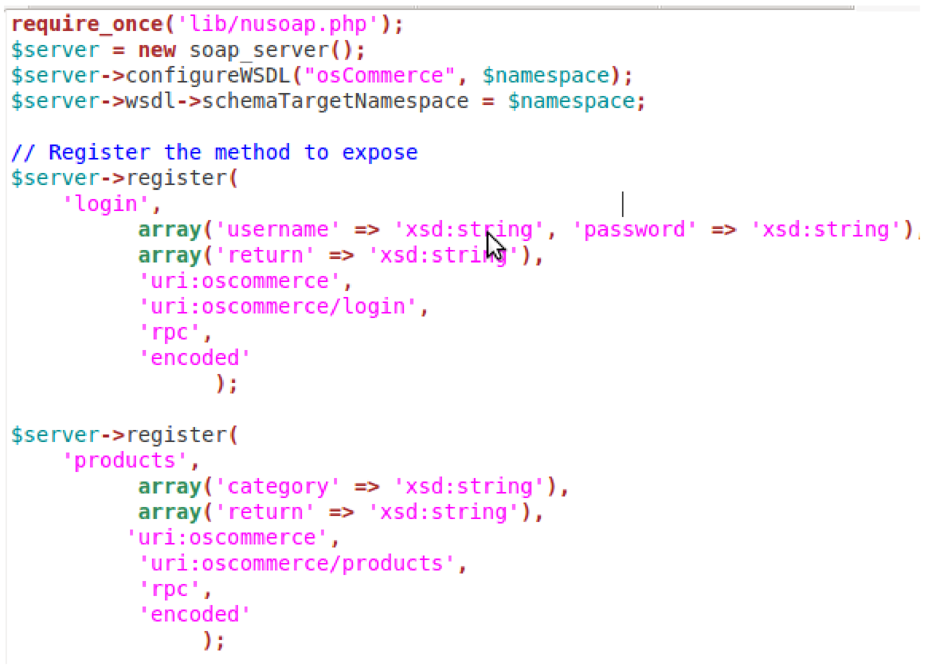


Figure 5.8 TeleWeaver components arrangement

TeleWeaver uses bundles to communicate with all the services that reside on this platform, all the services uses bundles, as discussed in Chapter Two that every service that is on TeleWeaver treat each other as if they resides on different platform.

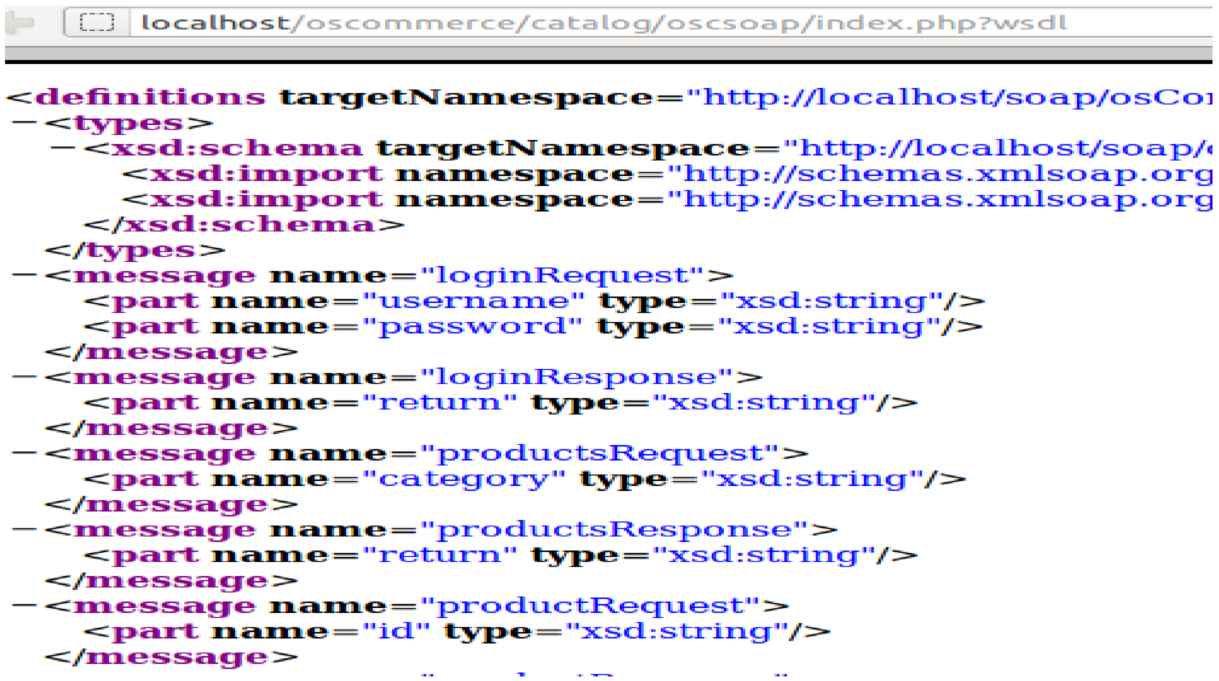
For this system there were several web services that were tried to integrate legacy applications with TeleWeaver middleware. As an example, Nu-SOAP web services was tried to link these legacy applications with TeleWeaver middleware.



```
require_once('lib/nusoap.php');
$server = new soap_server();
$server->configureWSDL("osCommerce", $namespace);
$server->wsdl->schemaTargetNamespace = $namespace;

// Register the method to expose
$server->register(
    'login',
          array('username' => 'xsd:string', 'password' => 'xsd:string'),
          array('return' => 'xsd:string'),
          'uri:oscommerce',
          'uri:oscommerce/login',
          'rpc',
          'encoded'
                );

$server->register(
    'products',
          array('category' => 'xsd:string'),
          array('return' => 'xsd:string'),
         'uri:oscommerce',
          'uri:oscommerce/products',
          'rpc',
          'encoded'
               );
```

Listing 5.5 Nu-SOAP Server

This is the Nu-SOAP server this part of the application registers the components of the legacy applications. This research has tried to register all the components of both legacy applications. Listing 5.5 illustrates the way of registering components into Nu-SOAP server using PHP. All the e-Commerce and Moodle components were registered to Nu-SOAP serve in order to integrate these applications into TeleWeaver. The research has discovered that these applications cannot be integrated into TeleWeaver middleware using Nu-SOAP, Because of the unavailable interfaces that support the integration of Nu-SOAP system we tried the RESTful Web service API. Listing 5.6 illustrates the system operation after registering those components on Nu-SOAP API.

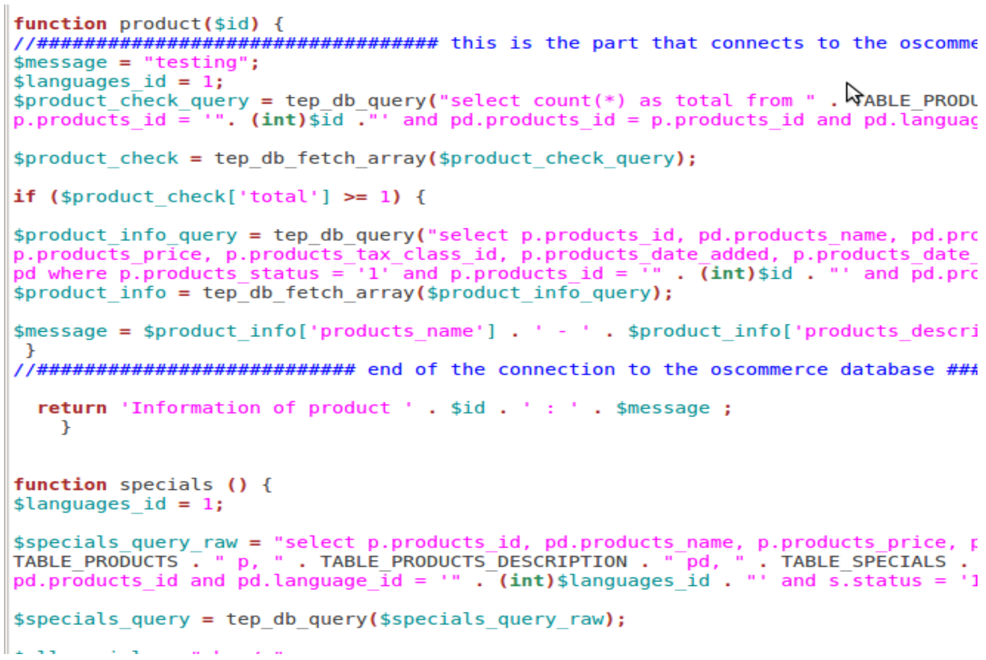


```
localhost/oscommerce/catalog/oscsoap/index.php?wsdl

<definitions targetNamespace="http://localhost/soap/osCo
- <types>
  - <xsd:schema targetNamespace="http://localhost/soap/
      <xsd:import namespace="http://schemas.xmlsoap.org
      <xsd:import namespace="http://schemas.xmlsoap.org
    </xsd:schema>
  </types>
- <message name="loginRequest">
    <part name="username" type="xsd:string"/>
    <part name="password" type="xsd:string"/>
  </message>
- <message name="loginResponse">
    <part name="return" type="xsd:string"/>
  </message>
- <message name="productsRequest">
    <part name="category" type="xsd:string"/>
  </message>
- <message name="productsResponse">
    <part name="return" type="xsd:string"/>
  </message>
- <message name="productRequest">
    <part name="id" type="xsd:string"/>
  </message>
```

Listing 5.6 WSDL information when using Nu-SOAP API

This above listing WSDL shows the operation that takes place after registering legacy applications components into the Nu-SOAP server. This operation shows all the vital operation that the client has to know about e-commerce web service such as URI of the service, name of the service, input and outputs of this legacy service. This is to alert the client about the service that is intended to be used.

However this operation is vital, effective and reduces the time of search for the services. This operation could not integrate our services into the middleware as stated in above section, due to unavailability of interface that supports Nu-SOAP connection. The next section will depict the way of connecting legacy service components with the database when using Nu-SOAP.

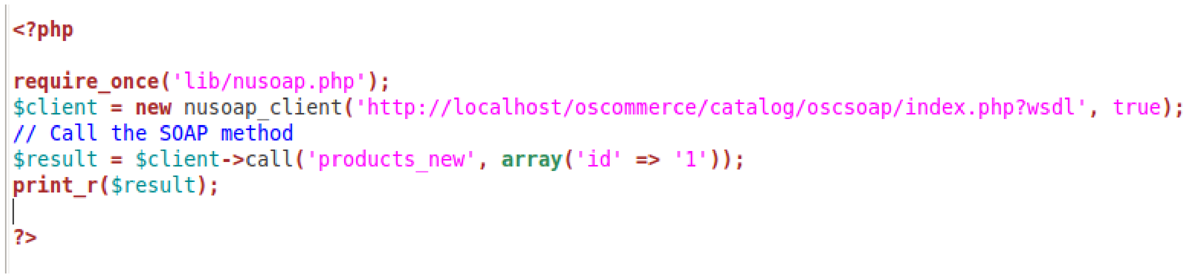


```
function product($id) {
//##################################### this is the part that connects to the oscomme
$message = "testing";
$languages_id = 1;
$product_check_query = tep_db_query("select count(*) as total from " . TABLE_PRODU
p.products_id = '". (int)$id ."' and pd.products_id = p.products_id and pd.languag

$product_check = tep_db_fetch_array($product_check_query);

if ($product_check['total'] >= 1) {

$product_info_query = tep_db_query("select p.products_id, pd.products_name, pd.prc
p.products_price, p.products_tax_class_id, p.products_date_added, p.products_date_
pd where p.products_status = '1' and p.products_id = '" . (int)$id . "' and pd.prc
$product_info = tep_db_fetch_array($product_info_query);

$message = $product_info['products_name'] . ' - ' . $product_info['products_descri
 }
//############################ end of the connection to the oscommerce database ###

  return 'Information of product ' . $id . ' : ' . $message ;
    }


function specials () {
$languages_id = 1;

$specials_query_raw = "select p.products_id, pd.products_name, p.products_price, p
TABLE_PRODUCTS . " p, " . TABLE_PRODUCTS_DESCRIPTION . " pd, " . TABLE_SPECIALS .
pd.products_id and pd.language_id = '" . (int)$languages_id . "' and s.status = '1

$specials_query = tep_db_query($specials_query_raw);
```

Listing 5.7 Connecting Nu-SOAP with e-Commerce Database

This above section explained how to register the service components when using Nu-SOAP API. This section demonstrates the way of connection API with each service component database. However as a main part of this research is to connect TeleWeaver with legacy applications, it is also important to connect API with each component database. When using Nu-SOAP connecting component with database is easy and flexible, the above list illustrates the connection of this application.

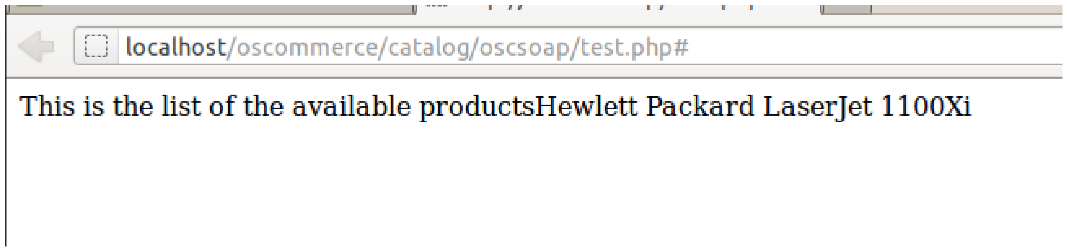
To see if the application does connect the database, the testing operation must take place. For the developers to prove that this component connects, the result of the test must give the positive result of the tested component. However, to test the connection, there is an operation that needs to take place. The Nu-SOAP test side must be written and it is where the developer checks if all services connect. This below list shows how is Nu-SOAP test side is created for testing these legacy application components



```
<?php

require_once('lib/nusoap.php');
$client = new nusoap_client('http://localhost/oscommerce/catalog/oscsoap/index.php?wsdl', true);
// Call the SOAP method
$result = $client->call('products_new', array('id' => '1'));
print_r($result);

?>
```

Listing 5.8 Nu-SOAP for testing services

All the service components have to be tested to see if they connect with the database. This is how to test the component connection when using Nu-SOAP. The Line 4 of the code illustrates the way of testing the each components connection. The line with $result=$client->call ('products_new',array()id=> '1'); is where the developer changes the prodct that she/he need to test, such as product, special and so forth. After changing the product in this part of the API then goes to browser and type the URI of the component that needs to be tested. The result will tell if the connection to the database is successful or not. Figure 5.9 shows the result when the connection is successful.

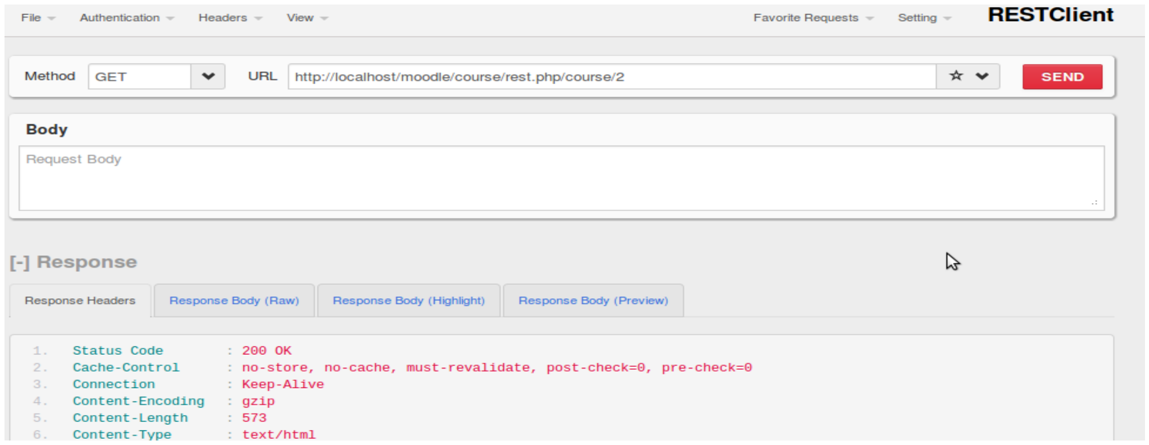


```
localhost/oscommerce/catalog/oscsoap/test.php#

This is the list of the available productsHewlett Packard LaserJet 1100Xi
```

Figure.5.9. Nu-SOAP Result

This above figure depicts the result of the available product in our e-Commerce store. This shows that the connection with the database is successful. Using Nu-SOAP API is flexible and easy, the integration part using this web service depends on the available interface that the middleware support.

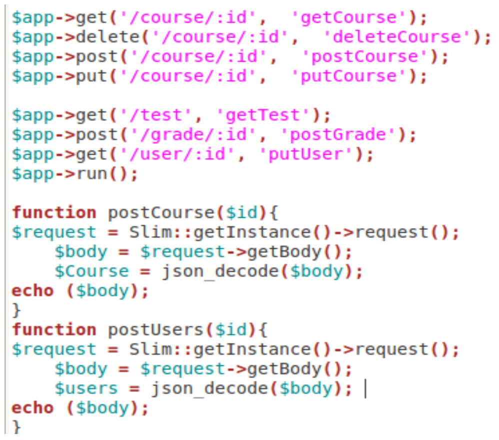
During the development of this research TeleWeaver middleware did not have the interfaces that support the integration using this web services. It was only discovered after the implementation of Nu-SOAP API that TeleWeaver does not yet support the integration of this web service, hence we integrated using RESTful web service API.



```
Status Code        : 200 OK
Cache-Control      : no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection         : Keep-Alive
Content-Encoding   : gzip
Content-Length     : 573
Content-Type       : text/html
```

Figure 5.10 Moodle connection with REAT API

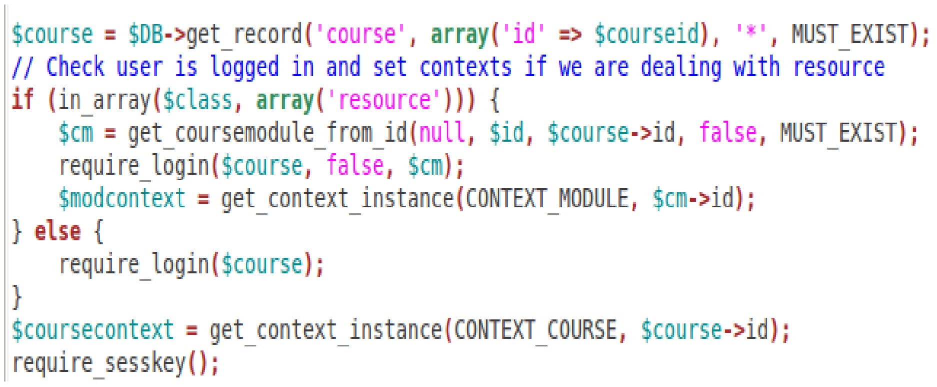
The Figure 5.10 illustrates the connection of rest API with Moodle components. This figure shows the positive results. The 200 OK on the response headers on the above figure shows that the connection is successful and there is no error on the connection. When the REST Client is not successful connected with component, it shows an error that says 404 not found. That clearly means there is an error. However rest client is mechanism that is used by the developer to test if the API is connecting with legacy applications components.



```
$app->get('/course/:id',  'getCourse');
$app->delete('/course/:id',  'deleteCourse');
$app->post('/course/:id',  'postCourse');
$app->put('/course/:id',  'putCourse');

$app->get('/test', 'getTest');
$app->post('/grade/:id', 'postGrade');
$app->get('/user/:id', 'putUser');
$app->run();

function postCourse($id){
$request = Slim::getInstance()->request();
    $body = $request->getBody();
    $Course = json_decode($body);
echo ($body);
}
function postUsers($id){
$request = Slim::getInstance()->request();
    $body = $request->getBody();
    $users = json_decode($body);
echo ($body);
}
```

Figure 5.11 Moodle registering methods on slim API

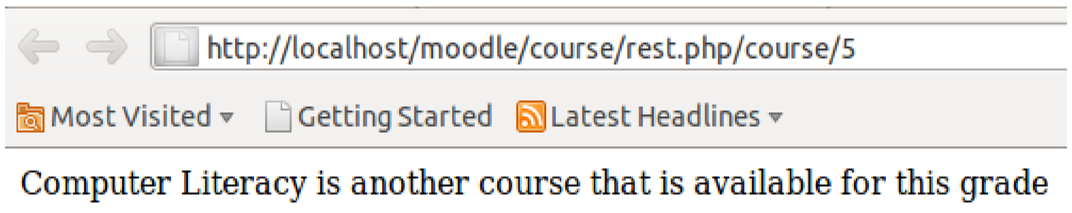
The above figure demonstrates how the developer registered the Moodle components on slim API. It also shows the methods that are used for these components. However the function for connecting these components is also demonstrated. This figure demonstrates how to post a method on REST API and the function of posting a component is also illustrated on the above figure.



```
$course = $DB->get_record('course', array('id' => $courseid), '*', MUST_EXIST);
// Check user is logged in and set contexts if we are dealing with resource
if (in_array($class, array('resource'))) {
    $cm = get_coursemodule_from_id(null, $id, $course->id, false, MUST_EXIST);
    require_login($course, false, $cm);
    $modcontext = get_context_instance(CONTEXT_MODULE, $cm->id);
} else {
    require_login($course);
}
$coursecontext = get_context_instance(CONTEXT_COURSE, $course->id);
require_sesskey();
```

Figure 5.12 Slim API connecting with Moodle database

The Figure 5.12 is shows the connection of Moodle database, specifically course component. This shows explicit details of how the database connection is done for Moodle services. The snap shot above illustrates how course module is connected to the database when using REST API.

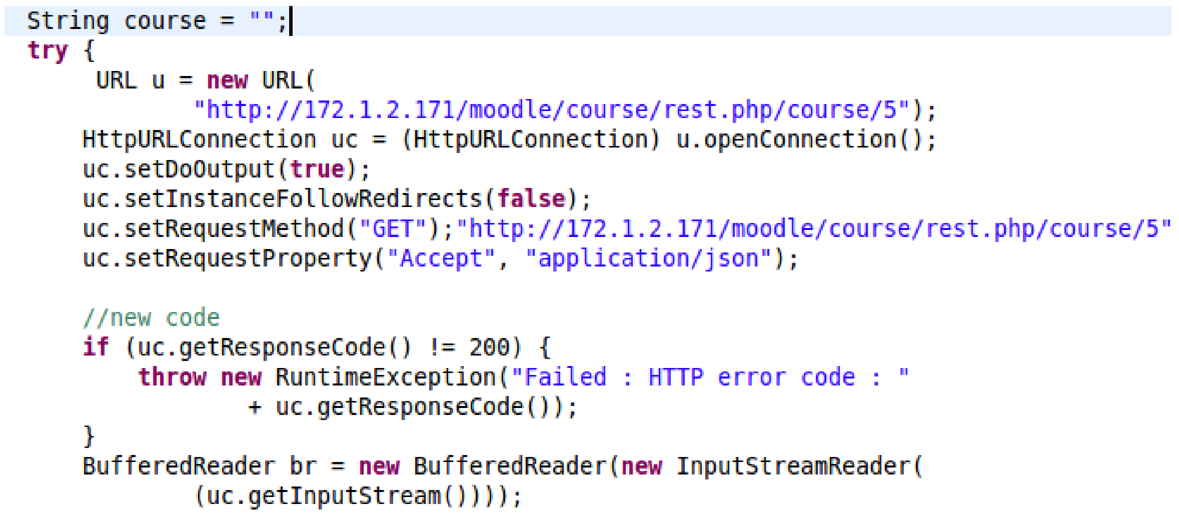


http://localhost/moodle/course/rest.php/course/5

Computer Literacy is another course that is available for this grade

Figure 5.13 Moodle response

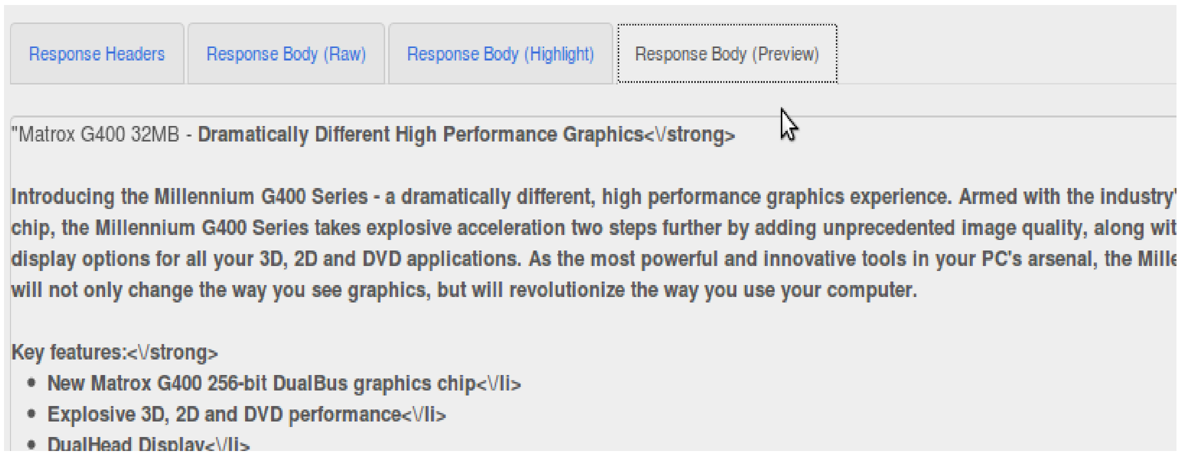
The above figure shows the results of Moodle components. This demonstrate that Moodle is successful connected with Slim API. This figure shows that REST API connects with Moodle database. These results show the courses that are available on Moodle database. However before the developer integrate Moodle application using REST API. The REST API has to be tested to confirm if it is connecting with the components database. Thereafter the developer can easily integrate that application into TeleWeaver middleware.



```
String course = "";
try {
     URL u = new URL(
            "http://172.1.2.171/moodle/course/rest.php/course/5");
    HttpURLConnection uc = (HttpURLConnection) u.openConnection();
    uc.setDoOutput(true);
    uc.setInstanceFollowRedirects(false);
    uc.setRequestMethod("GET");"http://172.1.2.171/moodle/course/rest.php/course/5"
    uc.setRequestProperty("Accept", "application/json");

    //new code
    if (uc.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : "
                + uc.getResponseCode());
    }
    BufferedReader br = new BufferedReader(new InputStreamReader(
            (uc.getInputStream())));
```

Figure 5.14 TeleWeaver connecting with Moodle Components

The Figure 5.14 demonstrates the code that has been used for connecting TeleWeaver middleware with Moodle components. This illustrates how Moodle components are called when using REST API. This works both remotely and locally. This has be proved and tested by the developer. However using this connection makes integration easy and flexible. When the developers or administrators seek to call a different method for Moodle, they only specify the component name and the method at highlighted code above, in the part where there is http and IP address. The important part when connecting TeleWeaver with legacy applications using REST API is the URI route of the component. A developer has to make sure that route is correct, one error on the route specification gives an error and the system will not connect. This above figure shows how Moodle is integrated into TeleWeaver middleware.



"Matrox G400 32MB - **Dramatically Different High Performance Graphics<\/strong>**

Introducing the Millennium G400 Series - a dramatically different, high performance graphics experience. Armed with the industry' chip, the Millennium G400 Series takes explosive acceleration two steps further by adding unprecedented image quality, along wit display options for all your 3D, 2D and DVD applications. As the most powerful and innovative tools in your PC's arsenal, the Mille will not only change the way you see graphics, but will revolutionize the way you use your computer.

Key features:<\/strong>
- **New Matrox G400 256-bit DualBus graphics chip<\/li>**
- **Explosive 3D, 2D and DVD performance<\/li>**
- **DualHead Display<\/li>**

Figure 5.15 The e-Commerce results using rest API

The figure above illustrates the successful connection of rest API and e-Commerce database. These are the result from e-Commerce components. This shows that the REST API is successful connected with e-Commerce database. The connection of REST API and the e-Commerce database makes work easier for TeleWeaver to connect with e-Commerce components using REST API. Table 5.1 highlights the details on REST service.

Table 5.1 An e-Commerce REST Client Firefox add on

Method	API route	Description
GET	Products/	This gets all the available products from the database
GET	Product/id	This retrieves the specific product-id from the database

POST	Product/id	This method allows the administrators to post a particular product to the database
DELETE	Product/id	The delete method deletes the specified route from the database
GET	Specials/	This retrieves all the specials from e-Commerce database
GET	Special/id	It retrieves the specified special-id from the database
POST	Special/id	This allows a developer to post a specific special-id on the database
GET	Reviews/	This retrieves all the reviews that are available on e-Commerce database

Table 5.2. Moodle REST Client Firefox add on.

Method	API route	Description
GET	Courses/	This retrieves all the courses that are available from the Moodle database

GET	Course/id	This method retrieves a specific course-id from the database
POST	Course/id	It allows a teacher to post a specific course to Moodle database
DELETE	Course/id	This method deletes the specified course-id from the database
GET	Tests/	It retrieves all the tests that are available from the database
GET	Test/id	This retrieves the specific course-id from the database
POST	Test/id	It allows the administrators to post a specific test to the database.
GET	Grade/id	It retrieves the specific grade from the database.
GET	Grades/	This retrieves all the grades that are available in database

The RESTClient add-on is where one tests if the API is functioning. This RESTClient grants various methods, where a developer can choose the method for each application component. This has been used in this research to test different components using different methods.

5.15 System implementation Technologies

This section explains the tools and the technologies used to integrate these two services into a TeleWeaver. This section describes the LAMP as a stack that was preferred for this project. It was chosen for this application because as stated in the previous sections that most the services that are deployed at Dwesa uses LAMP. The following section will give details of Linux platform which uses PHP component.

5.15.1 Linux platform

The current e-Services that are deployed at Dwesa are based on Linux as Operating system. Linux offers less maintenance than other operating system. Linux has grown as an operating system and a tool for personal and business use it has become more user friendly and more powerful as a back-end system (Christopher, 2010). Linux has proved to be viable, stable, and readily accessible to even those who don't consider themselves computer gurus. Linux was chosen for reliability, flexibility and cost-effectiveness. This platform was deemed as adequate for the deployment of the SLL due to its flexibility, robustness, effectiveness and the easy configuration and maintenance. Linux is a flexible, fast, secure operating system that is mostly used by developers. Linux has moved from being a specialty operation system into mainstream (Christopher, 2010). Linux has become a formidable operating system across the variety of business applications.

5.15.2 PHP Platform

PHP is used for deployment of the web applications (Cholakov, 2008). It is a popular web programming platform due to its simplicity, easy usage and use to learn (Cholakov, 2008). It is widely used tool for the development of dynamic web applications, it is easy to use, fast to learn and reliable programming language that offers easy configuration of applications. The e-Commerce, e-Learning and e-Health applications that are deployed in Dwesa they all uses PHP. PHP provides programmers with all the necessary tools to build dynamic Web applications from open-source software (Dave, 2008). Since most of Dwesa e-services have been developed using PHP. PHP is deemed to be an adequate language to use.

5.15.3 Apache web server

Apache is the web server that is recognized as a most popular web server, which supports plug-in modules for extensibility (Michele and Jon, 2007). Apache is a web server that directs the web browser into a resulting web page and knows how to process PHP code. Apache is a reliable server that is easy to configure. It supports several features such as PHP, Perl, python (Michele and Jon, 2007). It is recognized as the fastest web server and has a high performance (Dave, 2008). As most of the legacy applications being considered in this project are written in PHP, apache is web server is essential to have. This is where the test will take place if the service components are working or not.

5.15.4 MySQL

MySQL is a relational database server with support for several database engines (Christopher, 2010). It determines the how data should be stored in a database and it is design to be reliable and portable. Database is more important when developing and implementing a project, since undetected faults in these applications may result in incorrect modification or accidental removal of crucial data. Once the data is mistakenly modified, the error may propagate and lead to more data corruption if left undetected. Due to the benefits that MySQL offers, those are the reasons it was chosen as database for these applications.

5.15.5 Maven 2

Maven is a building tool that contains POM file and archetype (Maven, 2012). It helps developers to build projects, clean and debug, it notifies a developer when there is a problem and give the line where the problem reside. It creates an optimized JavaScript with XML file that contains the information of the application. Maven has eliminated the necessity for a developer to create command line procedures when writing code. It has maximized the productivity of developers by eliminating the time to check for an error by telling the where the problem is and what is needed to solve that problem.

The following section will give details on the adaptors that we use to integrate these legacy applications into TeleWeaver SOA Middleware.

5.16 Slim API

Slim is the API that has been used for this project due to its advantages. The main reasons for using Slim API in this project is because slim embraces the key characteristics HTTP protocol. Slim API supports GET, POST, DELETE, PUT methods, which are the key requirements for this integration project. Since Slim focuses on REST and it is a lightweight framework, it has been used in this integration project.

Slim is a lightweight framework that is used to build Restful API in PHP (Christophe, 2011). Slim API has a minimal code footprint, is easy to configure, and simple to use, hence it's called slim (Christophe, 2011). With Slim it's easy to connect the existing PHP functions to return data in response to URL request. "A route is a URL path defined relative to the slim base directory" (Andrew, 2010). There are several APIs that available for building a RESTful API.

5.17 Conclusion

This chapter has demonstrated how the integration of legacy applications into TeleWeaver works. The detailed discussion of the connection has been given. This involves the demonstration using the snap shots. The detailed discussion of rest API and REST Client add on and detail discussion of slim API. This has illustrated how TeleWeaver connects with third-party applications using restful web services. The next chapter addresses the essential matters of system testing, validation and evaluation of system's ability to meet the stated objectives.

6. Chapter 6: System Testing and Validation

The previous chapter has explained how the system was implemented and how it was developed. This chapter gives explicit details on how the system is tested, and the objectives of this chapter are to validate the system functionality and to achieve the last objective that is stated in Chapter One of this thesis.

6.1 Introduction

After the system was implemented, the next step was to test if the system works as expected. The testing was done to ensure that TeleWeaver communicates with legacy services. And also to ensure that all the system components function properly. The main users of the system were school teachers and art and craft member. This system was intended to assist the art and craft members to speed up the business process and to help teachers to improve the education system in rural communities. These rural ICT users were involved in the testing process to get feedback on the usability of the system and to validate the limitation presented when one uses stand-alone application and the benefits of integrating these applications into a middleware platform. Firstly, testing was to validate the limitations presented by using TeleWeaver interface to upload the products and information and the one presented by using a stand-alone services. The second part of testing was to check the flexibility and usability of the application integration services. The final test was to verify the adequacy of the integration system.

6.2 System Component Testing

As stated in Chapter Three, the developed system consists of various system components that perform different functionalities. Each of these components constitutes units in this system. However, this system is tested against the main objectives of the study. The testing is performed on each component of the system to check it against the stated measures. This is done to test the best way of integrating legacy services in to SOA middleware.

6.3 Testing Process

The testing process involves the validation and the verification of all components such as API connection with core service components of legacy application, GWT interface, and TeleWeaver connection with restful service API. As this system comprised of unit components, every component was tested against the stated measure on Chapter One. The next section discusses the testing of each unit of the system, starting from API testing up to TeleWeaver connection with third-party services.

The primary testing was performed to validate the communication between core service components and the slim API. The second testing was to test if the API can connect with the legacy application and if it can perform all the methods required by the administrator.

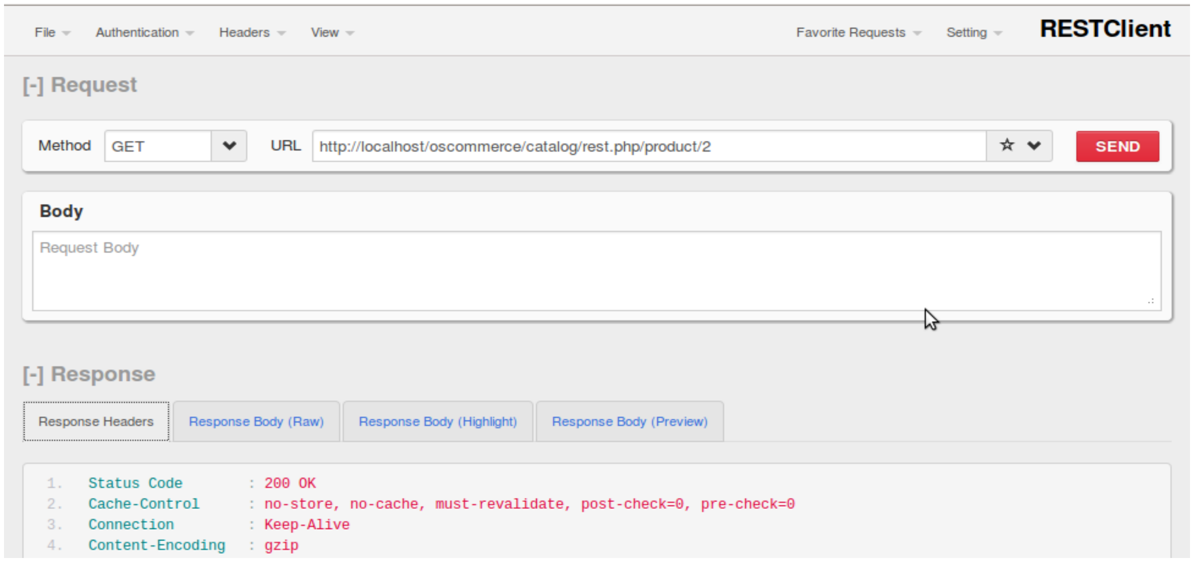
An explicit detail of testing protocols is discussed in the following section.

6.4 Testing the API

The Slim API in this system is utilized based on the requirements of this system, this system uses PHP scripting language and it requires a restful web services. Since TeleWeaver is JAVA based middleware, an interface that allows developers to integrate a PHP applications, was built during the course of this study. The Slim API in this system is provided by restful web services. The installation and configuration of the component was done for this component to function. The testing procedure of this component was done in two dimensions.

Firstly, testing was done to check the bugs and errors on API, this was done on Linux Edu-Buntu, using PHP language. Second testing was done to validate connection between slim API and the basic core components of the legacy applications. This was done using rest client add-on. In this dimension the important phenomenon is URI and route of each component. Using RESTful service, these above mentioned arenas are critical services in the development of REST API. However, an incorrect route or URI of a service brings error to the whole system. The API has several roles to play in this system, firstly it has to connect with every component of Moodle and e-commerce, and also it has to connect with TeleWeaver middleware.

To test if Slim API connects with legacy application components, REST Client has been used. The following listing shows how is this components tested.

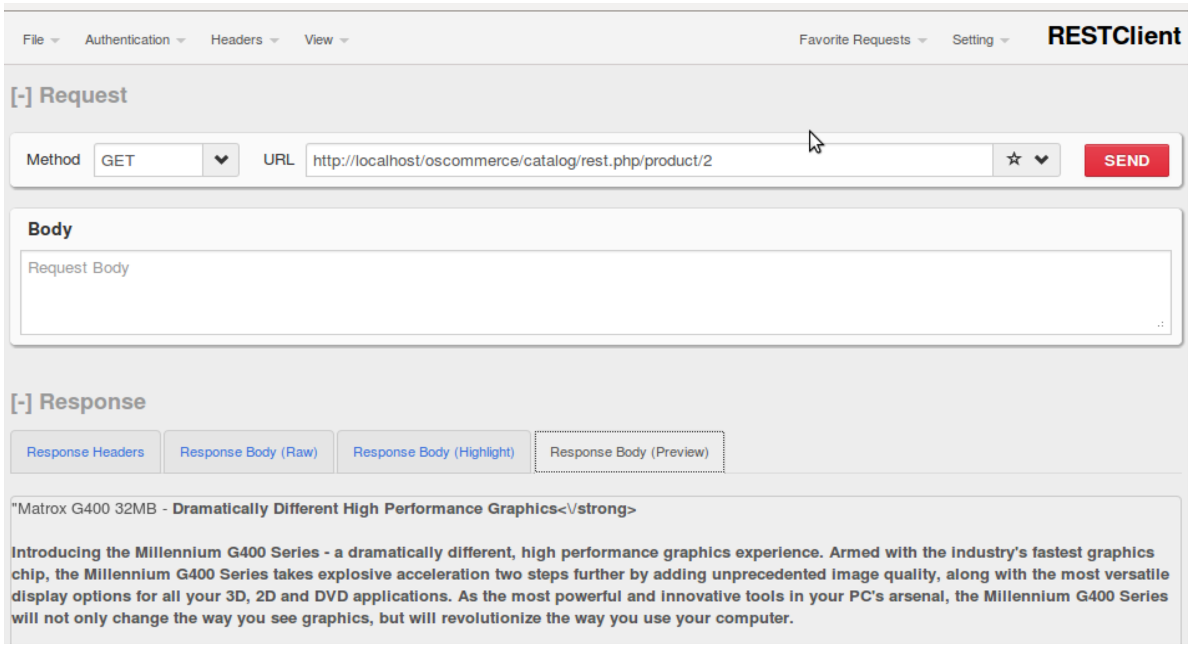


```
http://localhost/oscommerce/catalog/rest.php/product/2
```

```
Status Code       : 200 OK
Cache-Control     : no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection        : Keep-Alive
Content-Encoding  : gzip
```

Listing 6.1 Testing API

The above listing demonstrates the response of rest client. In the Response headers, the status code: 200 OK, validate the success of API, this means the Slim API is working. This means the URI and the route of the service that is invoked is OK or valid. The following listing illustrates the connection of API with the database of legacy application component.



```
http://localhost/oscommerce/catalog/rest.php/product/2
```

"Matrox G400 32MB - **Dramatically Different High Performance Graphics<\/strong>**

Introducing the Millennium G400 Series - a dramatically different, high performance graphics experience. Armed with the industry's fastest graphics chip, the Millennium G400 Series takes explosive acceleration two steps further by adding unprecedented image quality, along with the most versatile display options for all your 3D, 2D and DVD applications. As the most powerful and innovative tools in your PC's arsenal, the Millennium G400 Series will not only change the way you see graphics, but will revolutionize the way you use your computer.

Listing 6.2 Testing API connection with e-Commerce database

The above listing shows that Slim API can connect with the service that is called on URL of rest client add-on and the method that is used on request method. The methods are that declared on the slim API such as GET, POST, DELETE and UPDATE must correspond with the method on the rest client. When calling a get service on the rest client add-on, it must be checked if the service is declared a get method on the API. If a service is called as a post method on the REST Client, and on the API development is a get method, that connection will not be successful.

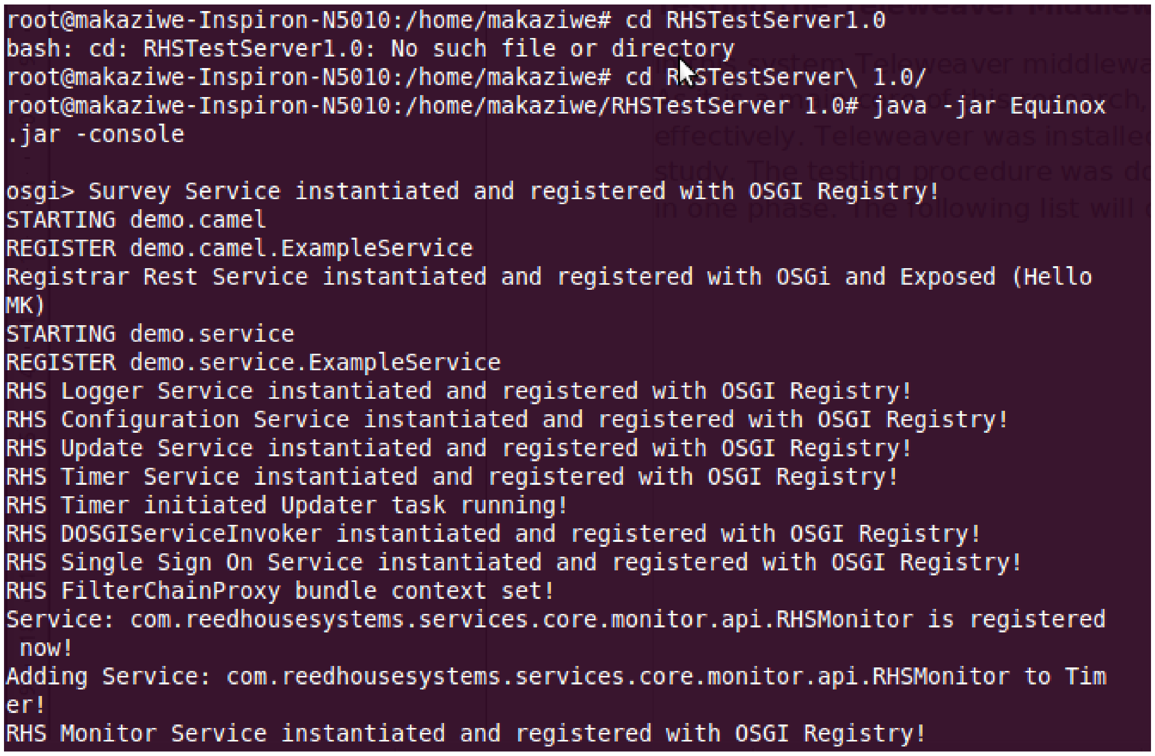
The data that is printed bold on the above Listing 6.2 is the information of the service that is on the database of that service that was invoked on the REST Client. This means the API connects with database of the service components of e-Commerce application and e-Learning application.

This displayed listing shows the successfulness of the API connection. However, this shows that the API was successfully configured, and it is well connected to each service that is declared. The in-depth of the configuration of this API is well documented on Appendix A.

It must be noted that when using a REST API, there are three fundamental aspects that are of value to the developers, which are Methods, Route and URI of the service invoked. If one misunderstands or incorrectly calls a service that uses GET method on the API, and the developer uses PUT method when invoking the service on REST Client, that gives error, because of the conflict of the methods. In this system these above listing shows that slim API is successful installed and configured and it can connect with the Os-commerce services. This API also has to connect with TeleWeaver Middleware. The following section explains on TeleWeaver testing.

6.5 Testing the TeleWeaver Middleware

In this system TeleWeaver middleware have to be installed and configured. TeleWeaver middleware has to function effectively since it is one of the key components of this thesis. TeleWeaver was installed as a deployment environment of this study. The testing procedure was done using Linux Edu-Buntu. It was done in one phase. The following figure demonstrates how TeleWeaver works.



```
root@makaziwe-Inspiron-N5010:/home/makaziwe# cd RHSTestServer1.0
bash: cd: RHSTestServer1.0: No such file or directory
root@makaziwe-Inspiron-N5010:/home/makaziwe# cd RHSTestServer\ 1.0/
root@makaziwe-Inspiron-N5010:/home/makaziwe/RHSTestServer 1.0# java -jar Equinox
.jar -console

osgi> Survey Service instantiated and registered with OSGI Registry!
STARTING demo.camel
REGISTER demo.camel.ExampleService
Registrar Rest Service instantiated and registered with OSGi and Exposed (Hello
MK)
STARTING demo.service
REGISTER demo.service.ExampleService
RHS Logger Service instantiated and registered with OSGI Registry!
RHS Configuration Service instantiated and registered with OSGI Registry!
RHS Update Service instantiated and registered with OSGI Registry!
RHS Timer Service instantiated and registered with OSGI Registry!
RHS Timer initiated Updater task running!
RHS DOSGIServiceInvoker instantiated and registered with OSGI Registry!
RHS Single Sign On Service instantiated and registered with OSGI Registry!
RHS FilterChainProxy bundle context set!
Service: com.reedhousesystems.services.core.monitor.api.RHSMonitor is registered
 now!
Adding Service: com.reedhousesystems.services.core.monitor.api.RHSMonitor to Tim
er!
RHS Monitor Service instantiated and registered with OSGI Registry!
```

Figure 6.1 A response of TeleWeaver SOA Middleware

The initialization of OSGI container in the terminal means TeleWeaver is successfully installed and is working. The above listing shows all the bundles that are installed on TeleWeaver. The line 4 of the listing elucidate protocol of starting TeleWeaver, the Java -jar Equinox.jar -console is the procedure of staring TeleWeaver. After installation of TeleWeaver to a directory, it needs to be restart to ensure if it is working. Once TeleWeaver show the OSGI line, it essentially means that TeleWeaver is successfully configured. The list of the installed bundles on TeleWeaver will appear. The first bundle initializes the registration of services in TeleWeaver, the second one keep the logs of the services. The third bundle instantiate the configuration of the services on OSGI registry. The forth bundle keeps the timer of these services that are registered and run on OSGI.

The DOSGI bundle is well discussed in Chapter Two. This service is installed on TeleWeaver to as a bundle that handle all the calls from different invoker, this services eliminates the procedure of knowing all the methods that have to be followed by a part service when it needs to make a call to a service that is stored in TeleWeaver. This eliminates the time spent on looking on each service requirement for communication.

The last bundle that appears when TeleWeaver is starting is Single Sing On service. This bundle is given here to grant credentials to the developers and administrator, so the person can be able to access other services without inserting credentials again, it authenticate the developer how who have a right to a program. If the person is granted an access to all the services, he/she will insert credential once; the other service will open automatically. This service is explained in Chapter Two of this research. However, if all these bundles appeared when starting TeleWeaver this ensure that TeleWeaver is successful configured.

The next section explains the main objective of this study, the core dimensions of this thesis and the connection of TeleWeaver with API.

6.6 TeleWeaver Connecting with Slim API

The main objective of this study was to integrate legacy applications with TeleWeaver Middleware. This means that the middleware has to connect with the third-party services to achieve the main objective. This section will ensure us if the main objective is achieved in this study. The above section explains the configuration and installation of TeleWeaver middleware, and the previous section discussed about slim API. This section is illustrating the major role of the mentioned platforms.

TeleWeaver is successfully configured, and the slim API is well installed and configured, both services have been discussed and tested and the result ensured us that these services are functioning as expected; therefore the connection between these two dynamic services is required to achieve the purpose of this study.

The connection of TeleWeaver and slim API was done, and it the results were positive, TeleWeaver does connect with third party applications. This was proved by requesting a service for TeleWeaver to Os-commerce component, and the component granted the response back to the developer and the administrator sitting on TeleWeaver. Also it was proven by requesting a service from TeleWeaver to connect with Moodle application, the Moodle component granted back the response. TeleWeaver connects with legacy application using restful services, which requires the present of slim API on top of each legacy application.

We have connected TeleWeaver middleware with Os-Commerce and e-Learning applications through slim API. TeleWeaver can connect with these two legacy application both locally and remotely. A developer can smoothly link Os-commerce with TeleWeaver, even if the developer is sitting on a different machine, which is remote connection. This was tested when a developer was sitting in different institution connecting with Os-commerce service on the other machine on other institution. This show that TeleWeaver can remotely communicate with legacy application, the phenomenon that is required for these two dimensions to connect is the IP address of the machine where Os-commerce and Moodle is installed and configured. Thereafter the system was working without any errors. This ensured the achievement of the main objective; however the developer can request information about a certain service that is e-Commerce or in Moodle application, using GWT. The service will bring back the result to a developer, when a developer seeks to upload and post data to a core service of the legacy application. The developer can just upload information using GWT and the information goes to the service of the legacy application and appears to the client side of e-Commerce application or Moodle.

This makes these legacy applications effective, extensible and sustainable, as a developer can update, delete and retrieve information about each service of the legacy application, remotely and locally. However, the integration of these services improves both education system and the commercial trend of the dedicated community.

As discussed in Chapter Two, GWT is developed in this system to allow developer to upload information and the item to the legacy applications. This service is developed using archetypes. This helps developers to speed up the process of coding, as archetypes use template that grant developer a clear and simple task. This is used in this system to avoid the spaghetti coding.

The GWT was tested to ensure if it connect with the API of legacy services, and if it allows the information to be uploaded. Information was specified on the GWT to the API of e-

Commerce application. The GWT has taken the information straight to the destination specified. The response from the legacy application service came back to GWT residing on TeleWeaver to show that the message was received. The testing of this interface elaborated the easy to use of this integration system. This was tested by several developers, colleagues and the administrators to test if this system is user- friendly. The following section discusses the usability of this system.

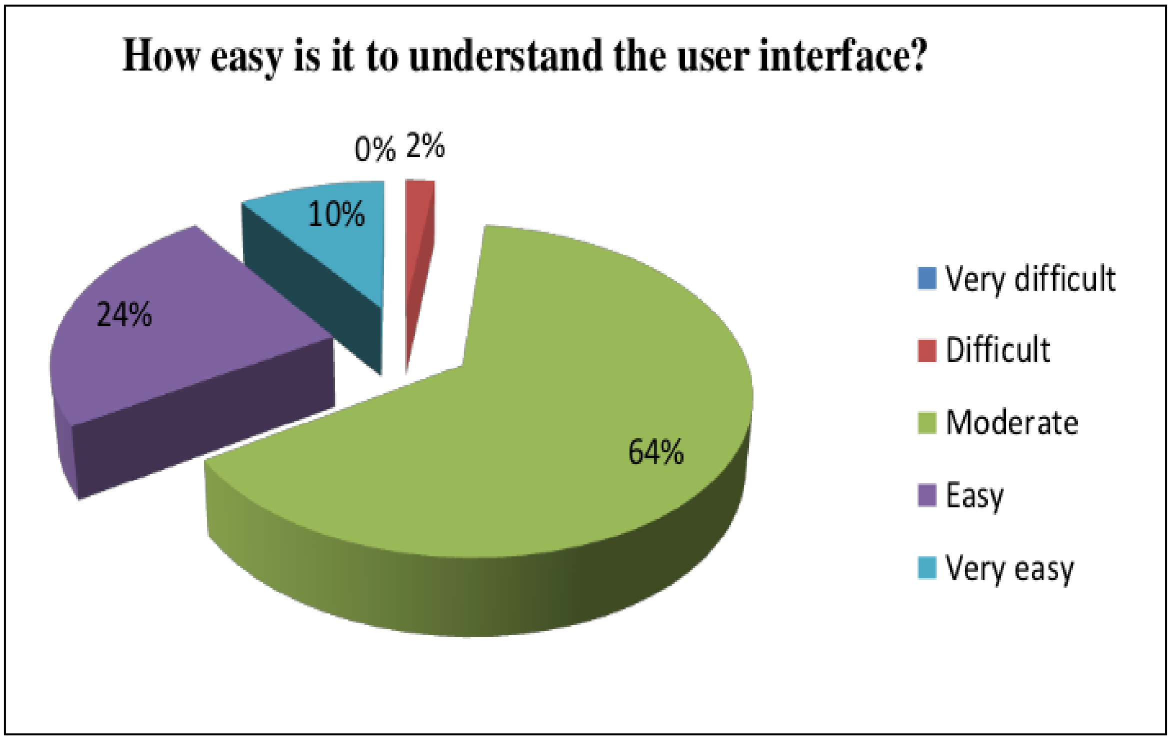
6.7 Usability testing

This system is developed for rural community members, who will use if frequently. As it is stated on Chapter Two that the Dwesa community members, they do not have a computer background, this system has to be easy to uses, effective and flexibility. The usability of the system was tested by different people who have different ICT background. The result of the system illustrated the easy-to-use of this system, all the people who were testing the system have not struggled, as every button on GWT is clear and simple. The positive result for the selected people shows the usability of this system. This brought upfront the argument that integrating stand-alone applications bring the effectiveness and sustainability to these applications. This was tested and proves in this study. However, updating and modernizing the components of these legacy applications was easy and simple comparing to the updating and maintenance of stand-alone application. Integrating these applications does not just help on improving the effectiveness of these applications; it also eliminates the amount of time spent on modifying the legacy applications components.

For usability testing we have tested both performance of the system and subjective metrics. For performance measures we included the success of the system, if the system was successful integrated. We have also tested the time and errors, user-friendly of the system. For the time we were testing the efficiency of the system, based on how much time did people spend on uploading and retrieving information from the legacy applications. This gives us the result that this system is efficient. Time is used to test how fast the system is. This was done by uploading information through GWT to the legacy application, and when we request the legacy application to retrieve the information back to the administrator. The system is very fast and effective when uploading and retrieving information. This illustrates that the system is effective and efficient.

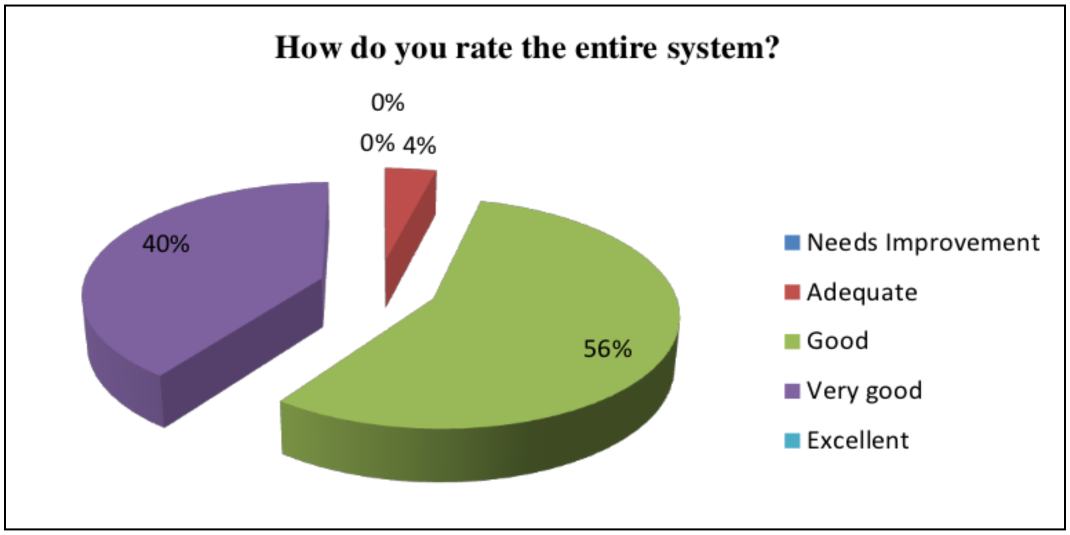
We also tested the subjective metric. Subjective measures include self-satisfaction, comfort rating and the report from the users. We asked 20 people from different perspective, from developers to ordinary people who know nothing about the programming, to test if the system is user-friendly. Below are the specific questions that we asked these different people who were testing the system.

Question1: How easy is it to understand the user interface?



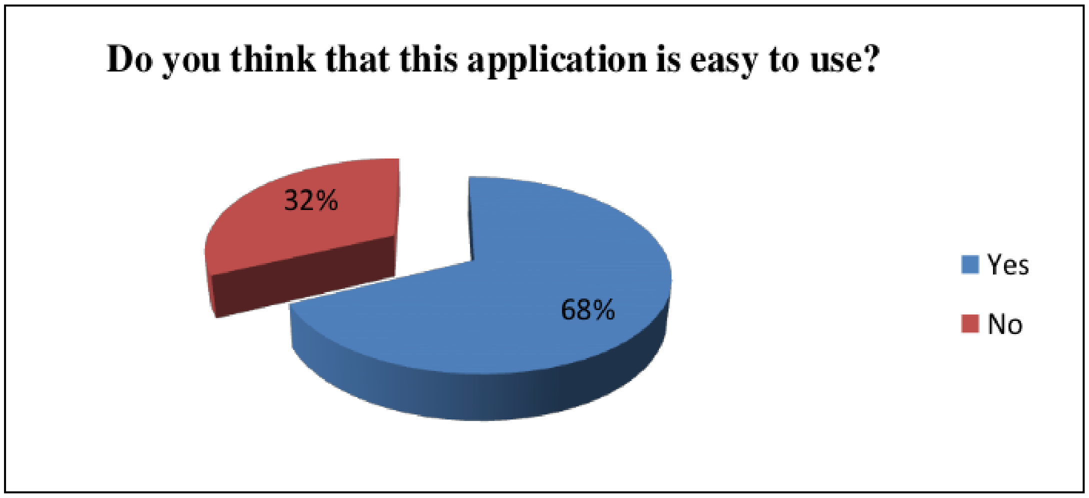
The pie chart above shows the results of the people that were involved on the survey. This is done to investigate the usability of the interface. The result are positive, more than a half of the people that were involved on the survey demonstrated that the user interface is moderate. As we were developing this system for MRA community, the user interface has to be easy and flexible. The other result shows that the user interface is very easy to use.

Question 2: How do you rate the entire system?



The above pie chat, gives an explicit details of the response of the people who were involved on the survey of rating the whole system. However the results depicted by the pie chat, shows that the system is good. The 56% of the people who were involved on the survey said that the system is good and flexible. The 40% of the people said the system is very good. There was no suggestion for improvement of the systems.

Question 3: Do you think this application integration system is easy to use.



This is the last question of the survey. This is to check if the whole application integration is easy to use. The different people from deferent background were asked this question as they were using this application integration system. The results of the survey show that the application integration system is easy to use. These results explain that, this integration system is effective, usable and flexible. The survey was done to test the usability of the system. The result shows the response of the people, who were testing the system.

We also tested the accuracy of the system, by checking how many mistakes the people made during testing of the system. This was done to determine the accuracy of the system. There were recoverable mistakes that the user during the system testing, but all the mistakes were recoverable mistakes.

For this usability testing we created a scenario whereby the ICT users sitting on TeleWeaver retrieve the data from the legacy application, also for the user to upload information to legacy applications. The ICT users we given and tested using this scenario to check the usability of the system. This testing method was to determine if the system met the main objective of this system, hence this study concluded by saying integrating legacy application makes these applications easy to use and effective.

In Chapter Four we have mentioned the functional and non-functional requirement of the system. However, the requirements were tested to validate if the system does what it is expected to do.

The first functional requirement was see if the TeleWeaver communicates with Os-Commerce and Moodle application. This was tested by using functional testing, to see if the TeleWeaver connects with these stated legacy applications. The ability of TeleWeaver to connect with these legacy application shows the system is functioning properly.

Based on the mentioned scenario this system has answered the functional requirements. All the functional requirements were met during the functional testing of the system.

The non-functional requirements of this system were met through conducting different testing methods. To see if the system is flexible and effective this was achieved through testing the functionality of the system and the how long does the system takes to respond to the user. The compatibility and the maintainability of the system was tested by checking how long does the system takes to recover from the errors and how many errors occurred.

Scalability and extensibility of this integration system was tested by adding more methods and route on the system to see if the system is scalable.

6.8 Validation and Evaluation of the system

The system interface is the major prime importance on directing the users of the system; hence it has to be easy-to-use, simple, effective and usable. The main factors that were used to increase the quality of interface is font side, simple words that are labels the system components, button and the background colours of the buttons. These factors were used to measure the quality of the interface. This was an important arena as it made the system flexible to all the testers.

All the units of this system were tested and they brought upfront positive results, this means that the objectives of this study were achieved. This answered the questions that were raised during the incubation of this project. The measures that were stated on Chapter One of this study were achieved by testing each system components against the stated measures.

The use cases scenarios in this project were used to exercise the vitality of system functional requirements and different data was used to test multiple components of this system. The main aim of the system was achieved. However, this procedure may be useful to other developers of SLL who would seek to integrate other stand-alone services to TeleWeaver middleware. The following section gives the overall discussion of this study.

The system was given to the testers to modify update the component of the legacy application using TeleWeaver interface, and using the normal stand-alone application interface, this was done to validate the flexibility of integrated application. The testers took lot of time in updating the system using stand-alone application protocol, compare to the amount of time they took when retrieving the information from the core service of existing application, when using TeleWeaver Interface. It was easy and fast to update the system when using TeleWeaver Interface. This concluded the study, by elaborating the effectiveness of integrating application to an SOA middleware.

6.9 Conclusion

The main aim of this chapter was to test the system and every components of the system if they meet the measures. This chapter has described the tests that were used in this research and how they were done. Multiple tests were carried in this chapter which includes the following.

The test of API functionality, Connection of slim API with core components of the legacy services, the connection of API with TeleWeaver middleware, the communication between TeleWeaver middleware and the Legacy applications, the integration of legacy applications into TeleWeaver middleware. All these tests were carried and discussed in this chapter. The following chapter provides the conclusion of the study and the research findings.

7 Chapter 7: Discussion and Conclusion

The previous chapter presented more details of how the system works and all the testing methods that were used to check if the system meet the stated requirement on chapter one. This chapter gives an overall discussion of the research and the findings of the research. This is the last chapter of this research.

7.1 Introduction

This project has presented the integration of legacy application into TeleWeaver SOA middleware. Several tests have been performed in previous chapter to test if TeleWeaver connects with legacy applications without any errors. The implementation chapter has elucidated how this integration is done in this study. The different tests have elaborated how the integration of existing application into TeleWeaver middleware can be used to enhance the performance and sustainability of ICT4D projects in Marginalized rural communities. This chapter presents the achievements, summary of project findings, the dynamic implication for future work and the overall discussion and conclusion of the whole study.

In Chapter One we set the context and direction of this project. This chapter concludes the dissertation by summarizing the work done and presents the future work. We present the primary contribution that we have made, and discuss problems encountered during this research and conclude by exploring the future work.

7.2 Summary of the thesis

The main aim of this research is to integrate legacy applications into TeleWeaver SOA Middleware. The research context in which this work took place is ICT4D. The motivation of developing this system under SLL intervention is that, SLL has developed several applications for marginalized rural community to improve the lifestyle of these communities using ICT, while these applications have been developed under ICT4D, it has been discovered that these legacy applications are unsustainable and not effective as expected. Due to the complications of maintaining standalone applications, it has been discovered that if these applications can be integrated into SOA middleware, the existing services can be flexible and effective, hence this system have to be developed to assist these application to deliver as expected, also to be flexible, effective and sustainable.

The proliferation of ICT4D interventions is the proof that ICT is an enabler for development. The ICT for community development has played a major role in improving the life of marginalized rural communities. The ability of these communities to use Internet without hindering is a testimony to community that ICT is enabler for community development. The role of ICT4D has to be viewed and understood in the light of different paradigms used to reduce poverty in marginalized rural areas also to eliminate the digital divide. Not long ago the marginalized rural areas such as Dwesa, were separated, not only physical separated but they have been separated from information, which leads these areas to be disadvantage areas. With the help of ICTs the community development managed to improve and enabled development of several applications that allow these communities to access information and know what is going on globally. This has liberated these communities. However, people from Dwesa can perform various tasks using the Internet, this shows the development on the community, hence ICT is no longer known as a computer study, but as an eye opener and enabler for community development. Siyakhula Living Lab has been using ICT to assist and improve these impoverish areas. Integration of legacy applications in SOA middleware is helping SLL to improve the legacy application and to enhance the standard of living in rural communities. This is encouraged by making ICTs applications effective and sustainable.

The role that Siyakhula Living Lab has performed on developing marginalized rural communities is an enormous role. This is elaborated by the ability of these communities on using computers and making use of the Internet platform. SLL intervention has performed training for community member, now the community members who were attending the training lesson, have computer literate certificates that presents their well-being of using computers.

The SLL has used ICT4D to develop marginalized communities, however developing the numerous stand-alone applications with a goal of enhancing the standard of living for these MRA's, present the eagerness of this intervention on developing these communities and to reduce the rate of poverty and enhance the knowledge system in these areas.

This project has been developed with aim of making these legacy services effective, flexible and sustainable. SLL has developed several applications for rural ICT users and there was need to integrate these existing services into SOA middleware for these to be effective and reusable. This can be used to reduce the cost of maintaining these services and the time spend on updating and modifying the applications. Due to low programming background and scarce skill of using computers in these areas, the developed applications have to be easy to use, cost-effective and sustainable: hence with application integration system, maintaining existing e-services is easy and flexible.

This is system was designed to enhance legacy applications, to make these e-Services effective, cost-effective and sustainable also to minimize the cost of maintaining these applications and time spent on modifying these services. This was done after investigating, through literature reviews, various funding methods, brainstorming with developers, meeting with school teachers and art and craft members.

The results of the literature survey show that standalone legacy applications are hard to maintain and they are not flexible to changes, they do not satisfy the end user' as they cannot meet the everyday changes. The results show that SLL possesses lot of standalone applications which presents some challenges in-terms of effectiveness and sustainability. Therefore, this application presented in this thesis exhibits the core components and layers through which these legacy applications are integrated on. The architecture of the system which is presented on Chapter 4 has elaborated the integration pattern.

7.3 General Discussions

The system was tested to see if it meets the stated objectives. This was done through the conduction of different testing methods that are stated in the previous chapter to answer the different questions. All the mentioned research questions were answered in different chapters of the thesis.

We have found all the answers for the stated research questions. In the course of the research we have found out that the main advantages of integrating legacy applications into SOA middleware is to make applications effective and maintainable and to meet today’s business needs. These are the benefits of integrating legacy applications into SOA middleware.

The second question of this research was, how best to integrate legacy applications into SOA middleware. This research has provided the answer to the second question of this research, the best way of integration legacy applications into SOA middleware is through integration using web services depending on which application that the developer seeks to integrate. If the developers want to integrate the PHP application then restful web service and Slim API is the recommended API to use. It is easy, flexible and user-friendly API that makes applications to be smoothly integrated to the SOA middleware. This also depends on the interfaces of the middleware. If the middleware that a developer seeks to integrate applications on supports PHP interfaces using restful web service and Slim API or adapter it is an easy way of performing fast integration process. In Chapter Two of this study we have discussed the different ways of integrating legacy applications into SOA middleware. We have tried and tested the different techniques of integrating these legacy applications into SOA middleware. We have tried and developed a system using Nu-SOAP adapter, after developing and tested we went back and changed the prototype due to the complications we come across when we tried to integrate using Nu-SOAP service, hence we concluded this study by recommendation of saying restful web service is the best techniques to integrate legacy applications into SOA middleware.

The last question is on comparing the different techniques of integrating legacy applications into SOA middleware, the following paragraph will answer that question.

In this research we have conducted different techniques of integrating existing systems into SOA middleware. In Chapter Two we have discussed the different integration method, and through the system development we have discovered that using XML-RPC and Nu-SOAP services is easy but it needs an adapter model where these legacy applications will use in order to be integrated into SOA middleware. These techniques can also be used depending on the interfaces that are presented by the middleware. This research has discovered that the RESTful service is easy to use, as creating its adapter models is easy and flexible. So it is recommended that using RESTful web service is flexible and easy, as it allows smooth integration of PHP legacy applications.

The objectives of the thesis were met through conducting different research techniques. Each chapter in this research has addressed various objectives of the research. The following section gives more details on how the objectives were addressed in this research.

7.4 Addressing the research objectives

The main objective of this study is to integrate legacy applications both e-Commerce and Moodle applications into TeleWeaver SOA middleware. The first objective was to develop the system that will integrate these legacy applications into SOA middleware. This objective was achieved by developing a working system that allows smooth integration of these stated legacy applications into TeleWeaver SOA middleware. This was discussed in Chapter Five of this research where the researcher developed a working system and tested the functionality of this system on Chapter Six. These two chapters achieved the main objective of this study.

The sub-objectives of this research were also meet in this research. The first sub-objective was to identify the benefits of integrating legacy application into SOA middleware; this was answered in Chapter Two of this research through discussion. In Chapter Two we discussed the benefits of integrating legacy applications which are, allowing the re-usability, flexibility and effectiveness of the applications.

The second sub-objective was to investigate the integration and application modernization techniques; this was also answered in chapter two of this thesis. Through a literature review we have discovered the integration and application modernization techniques that can be used to integrate legacy application in a smooth and effective manner.

To achieve sub-objective three of this research of how to best integrate legacy application specifically Os-Commerce and E-learning applications into TeleWeaver middleware, these objectives were met through development of the working system which is clearly stated in Chapter Five of this thesis.

The second last sub-objective was to investigate the web service techniques, API and libraries that can allow smart integration of the existing applications. This objective was achieved through using different method, first method was to do literature review and the second was to develop the working system. Using these two stated methods it is easy to identify the best web service technique and API that allows easy integration of legacy applications into SOA middleware. Chapter Two and chapter five of this research give a clear answer to this objective.

To implement and to field test the systems which incorporates both the validation of system against the states measures, and limitations of stand-alone services and usability of this system. These objectives were achieved in developing the well-functioning system which is Chapter Five and testing the functionality of this integration project which is Chapter Six of this thesis and evaluating the effectiveness of this system is achieved on Chapter Six of this thesis.

This system is designed to enhance the legacy applications that are developed SLL intervention for marginalized rural communities of the Eastern Cape in South Africa. Base on the requirement analysis on Chapter Three, a system was designed, implemented and tested to demonstrate the benefits of integrating these e-services into SOA middleware.

The second objective was to investigate and understand the integration and application modernization techniques as to determine which integration procedure will be suitable for SLL legacy applications. Through a literature survey, a series of integration procedures were investigated in Chapter Two and found that TeleWeaver middleware which is discussed in chapter two, have several interfaces that allow developers to integrate their applications on it.

The third objective was to identify the best way of integrating legacy applications specifically e-Commerce and Moodle application's into TeleWeaver middleware. The investigation was done through literature review and study of TeleWeaver middleware , it was discovered that the best and suitable criteria of integrating these existing applications into TeleWeaver was to develop RESTful web services which will use Slim API to allow connection between TeleWeaver middleware and stated legacy application. In this study we found that both of these legacy applications are developed using PHP scripting language, hence we developed restful web service API to concede the connection. This is found as the best and flexible way of integrating legacy applications into TeleWeaver middleware.

The fourth objective was to investigate the web service technology, API and libraries that can be utilized to allow smooth integration of existing disparate applications into SOA middleware. We have found that using restful web services is the beginning of flexible way of integrating legacy application. Using Slim API grants developers a smooth integration platform. Through the development of the integration system this research discovered that slim API grants a flexible platform of connecting SOA middleware platform with legacy applications. In this manner the integration system is easy, effective and flexible. This led to sustainable and flexible integration system.

The fifth objective was to implement the integration of these legacy applications. This objective was achieved by the use of dynamic scenarios presented in Chapter Five. Chapter Five demonstrated the implementation of the integration system.

The last objective was to evaluate the usefulness of the integration of legacy application and validate the adequacy of this system. This objective was achieved by testing the adequacy and flexibility of this system. Chapter Six of this thesis has elucidated the usefulness of the integration system. Several RHS developers have tested flexibility of this system. In this manner the objectives which were the guideline of this study were achieved in this research.

All the questions and measures that were stated in Chapter One were met and answered. All the objectives, questions and aims that are in the scope of this study were achieved.

7.5 Future Work

The integration of legacy applications into TeleWeaver SOA middleware is limited due to interface presented by TeleWeaver. This is a loophole that needs to be addressed in the future improvements of the integration system. Due to the proliferation of SLL at which deferent projects are developed per year, there is a need for several protocol or Dynamic Service Invoker or interfaces that allows legacy application to be integrated into TeleWeaver middleware without difficulty. The recommendation is to have a super services invoker on TeleWeaver which allows smooth communication between legacy services and TeleWeaver. Another technical challenge that rose during the course of the study was to have TeleWeaver installed at RHS side only, which give critical problem when developing integration system, as it uses RHS credentials. Having super service invoker will provide support to every interface of the application such as XML-RPC, Nu-SOAP, SOAP or REST. This will make an easy integration to all the legacy applications regardless of which language does the application use. Current TeleWeaver SOA middleware support REST and SOAP interfaces, the extension of TeleWeaver interfaces will provide an easy integration.

7.6 Overall Conclusion

This thesis has described the integration of legacy applications into TeleWeaver SOA middleware. This has described the integration platform and the best way of integrating the stand-alone applications under the SLL context. Several scenarios have been demonstrated to explain the integration of the legacy application into TeleWeaver. The various ways of integrating legacy application has been discussed and the recommended way of integrating existing application has been explained. The interfaces and the layer which integration took place have been elaborated. The way which Slim API sit on top of these applications and the way slim connects with core service components of legacy application is discussed. The flexible standards which slim connect with TeleWeaver middleware has been elucidated by several scenarios. This project was developed based on the open source platform to grant full ownership to the administrators. This project has demonstrated how to integrate legacy applications; this will give guidelines to developers who will integrate other legacy applications into SOA middleware.

References

1. Aaron, R. (2008). Restful Web Services. Fort Lewis College, Durango.
2. Aberdeen Group. (2009). Analyst Insight, Legacy Modernization in a Tough Economy: Cost and improving Agility. Swissotel. Chicago.
3. Aberdeen Group. (2006). The Legacy Application Modernization Benchmark Report: Transforming MainFrame, AS/400 and Unix Applications in an SOA World. Aberdeen Group inc. U. S. A.
4. Alcatel-Lucent. (2006). XML Web Services & APIs. Available from: http://enterprise.alcatel-lucent.com/?product=XMLServices&page=overview (Accessed 12-October 2010).
5. Alistair, C. (2000). Writing Effective Use Cases, Humans and Technology in preparation for Addison-Wesley Longman. Cockburn. Canada.
6. Andrew J. A.(2010). Using The Slim PHP Framework for Developing REST APIs. Available from:http://codingthis.com/programming/php/using-the-slim-php-framework-for-developing-rest-apis/. (Accessed on 03-September-2012)
7. Apache. (2010). ApacheApache CXF CXF. Available from: http://cxf.apache.org/distributed-osgi.html. (Accessed on 12-august-2012)
8. Autodesk (2012). Digital Prototyping: Design, Manufacture, and Market the World's Best Product. Available from: http://usa.autodesk.com/digital-prototyping/ (Accessed on: 20-September-2012)
9. Autodesk Manufacturing. (2012). Digital Prototyping. Available from: http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=16695831 (Accessed on 15-August.2012)
10. Borland Software Corporation. (2009). Effective Requirements Definition and Management. Available from:http://www.borland.com/resources/en/pdf/solutions/rdm_whitepaper.pdf .(Accessed on 25 July 2012).
11. Bradley, M. (2012). Wireless and networking. Available from: http://compnetworking.about.com/cs/webservers/g/bldef_apache.htm. (Accessed by 12-September-2012)

12. Bradley, A. (2008). SOA Scenario: Pattern and Guidelines for SOA Starting with SOA and Moving to Advanced SOA. Gartner Researcher.
13. Carroll, J.M, (2000). Mental Models in Human Computer Interaction. Available from: http://www.ntrs.nasa.gov/.../19890068859_1989068859.pdf .(Accessed on: 20-06-2012)
14. Cerami, E. (2002). Web Service Essentials: Distributed Applications with XML-RPC, SOAP, UDDI &WSDL. 1St edition. O'Reilly Media, Inc. United State of America.
15. Charles, D.(2003). How to plan a road map for application modernization. Available from:www.computerworld.com/s/article/86137/How_to_plan_a_road_map_for_application_modernization. (Accessed on 01-September-2012).
16. Charles Dickerson.(2003). How to plan a road map for application modernization. Available from:Http://www.computerworld.com/s/article/86137/How_to_plan_a_road_map_for_application_modernization. (Accessed on 01-September-2012)
17. Christophe, A. (2011). RESTful services with jQuery, PHP and the Slim Framework. Available from: http://codingthis.com/programming/php/using-the-slim-php-framework-for-developing-rest-apis/.(Accessed on 01-September-2012)
18. Christophe, C. (2011). Restful services with Jquery, PHP and the Slim Framework. Available from: http://coenraets.org/blog/2011/12/restful-services-with-jquery-php-and-the-slim-framework/ .(Accessed on 01-September-2012)
19. Christopher, N. (2010), Linux Bible 2010 Edition: Boot Up to Ubuntu, Fedora, KNOPPIX, Debian, openSuSe and 13 Other Distribution. 2010 edition. Wiley Publishing. Canada and India.
20. Clifton, M. (2003). What Is A Framework ? Available from: http://www.codeproject.com/Articles/5381/What-Is-A-Framework. (Accessed on 12-September-2012).
21. Crugnalo, A. (2006). Using Web service With Flash and Nusoap. Available from: http://www.sephiroth.it/tutorials/flashPHP/webService/ (Accessed 15 April 2010).

22. Dalvit, L.; Terzoli, A.; Muyingi, H. & Thinyane, M. (2007). The deployment of an eCommerce platform and related projects in a rural area in South Africa. In International Journal of Computing and ICT Research, Vol 1, No 1, p 9-17.

23. Daryl, K., and Eamonn, G. (2004). Uses Case Requirement context, 2edition, Published by Pearson Education. United Stated of America.

24. Dave, M. (2008). Php3: Programming Browser-Based Applications with Php,1st McGraw-Hill Professional ©1999 available http://dl.acm.org/citation.cfm?id=540803

25. Davey, B. & Cope, C. (2008). Requirements Elicitation – What‟ s missing? In the Issues in Informing Science and Information Technology, Journal Vol 5, p1547-5840, Informing Science Institute, 131 Brook hill Court, Santa Rosa, CA, 95409, USA

26. David, B., and W3C Fellow. (2004). Web Services Architecture; W3C working group note 11 February 2004. Available from: http://www.w3.org/TR/ws-arch/. (Accessed on. 23-August-2011).

27. David, B., and W3C Fellow. (2004).Web Services Architecture; W3C working group note 11 February 2004. Available from: http://www.w3.org/TR/ws-arch/. (Accessed on. 23-August-2011).

28. Dickerson, C. (2003). How to plan a road map for Application Modernization. IDG inc. Available from: http://www.cumputerworld.com/s/article/ 86137/#

29. Donald, C. (2010), Brainstorming, Big Dog and small dog's performance juxtaposition. Available from: http://www.nwlink.com/~donclark/perform/brainstorm.html. (Accessed on: 01-august-2012)

30. Dymond, A., and Oestmann, S. (2003). A rural ICT Toolkit for Africa. Worldbank/InfoDev. Available from: http//www.infodev.org/projects/telecommunications/351africa/ruralICT/toolkit.pdf

31. Ellen,S., Stephen, F., Aaron, W., Robert, L and Arnold, R. (2005). Linux in a Nutshell (In a Nutshell (O'Reilly)), O'Reilly Media. Published by ACM. New York.

32. E-European, (2006). E-European Justice. Available from: https://www.e-justice.europa.eu/content_judicial_systems-14-en.do. (Accessed on 10-September-2012).

33. François, C. (2010). What Is System Architecture? [Online]: http://www.wisegeek.com/what-is-systemarchitecture.htm. (Accessed on 18 July 2012)

34. Freivald, J. (2006). iWay SOA Middleware: An Agile framework for Fast, Flexible, Low-Risk services Deployment. iWay inc. Unite Sate of America.

35. Gilkey, H. T. (1960). "New Air Heating Methods", New methods of heating buildings: a research correlation conference conducted by the Building Research Institute, Division of Engineering and Industrial Research, as one of the programs of the BRI fall conferences, November 1959., Washington: National Research Council (U.S.). Building Research Institute, pp. 60.

36. Glenn, J. Browne, Michael B and Rogich. (2004). http://mesharpe.metapress.com/app/home/contribution.aspreferrer=parent&backto=issue,10,10;journal,45,48;linkingpublicationresults,1:106046,1 (Accessed 20-July 2012)

37. Gudgin, M.; Marc, H.; Lafton, Y.; Mendelsohn, N.; Moreau, J. & Nielsen, H. (2007). SOAP version 1.2 part 1: Messaging Framework. 2Nd edition. World Wide Web Consortium. Available from: Http://www.w3.org/TR/Soap12-part1/

38. Harris, R.W. (2004). Information and communication technologies for poverty alleviation. Kuala Lumpur: UNDP's, Asia-Pacific Development information Programme.

39. Helsinki Living Lab. (2007). What is a Living Lab? Available from: http://www.helsinkilivinglab.fi/node/162. (Accessed 20 July 2010).

40. Howard, C. (2009). Oracle IT Modernization Series: Why Modernization? An Oracle White Paper. Published in U.S.A. Available from: www.oracle.com/us/technologies/modernization/027006.pdf. (Accessed on 01-September-2012).

41. Jack, V. (2007).Best practices for OSGi development in enterprise Java integration. Available from: http://www.soa.com/Best-practices-for-OSGi-development-in-enterprise-Java-integration.html. (Accessed on: 12-September-2012)
42. James, D. (2012). E-Commerce Infrastructure:Middleware in Web 2.0. Avilable from: http://www.slideshare.net/jamesdellinger/mecb-middleware-and-web-20-project-2012. Accessed on: 10-October-2012.
43. James, T.; Lahde, K.; Naidoo-Swetternham, T and McKay, K. (2011). A Practical Approach to ICT for Development: Perspective from SAFIPA programme. Pretoria. CSIR Meraka Institute, South Africa.
44. Jonathan, B., Sean, K,. and Bobby, S,. (2005). A NEW SERVICE-ORIENTED ARCHITECTURE (SOA) MATURITY MODEL. Available from: www.omg.org/soa/Uploaded%20Docs/SOA/SOA_Maturity.pdf . (Accessed on 20-march-2011).
45. Josuttis, M. N. (2007). In SOA in practice: The art of distributed system design. California: O‟ Reilly.
46. Kuntal, G. (2011).Developing a web service and consuming it withJ2ME Client. Available from : http://www.ibm.com/Developing a web service and consuming it with J2ME Client.pdf. (Accessed on 20-September 2012).
47. Karen, J.B. (2011). Stepping Up to The Application Integration Challenges. 1St edition Oracle Inc. New York.
48. Kumar, P., Perreira, M., Vaidya, P., Vosseler, F. & Peltz, C. (2006). Moving from point-to-point integrations to SOA-based integrations. Hewlett-Packard Technical Information.
49. Laurent, S.; Johnson, J.; Winer, D & Dumblil, E.(2001). Programming Web Service XML-RPC . 2Nd Edition. O'Reilly &Associates. United State of America.
50. Laurent, R.(2010). SOA Middleware: SAP NetWeaver PI as a Central Integration Hub. 1St edition. SAP AG. India.
51. Living Labs in Southern Africa (2009). Overview. Available from: http://llisa.meraka.org.za/index.php/Overview (Accessed 20 July 2010).

52. Makamba, M. and Thinyane, M.P. (2011). Implementation of an Adaptor Component to Integrate Legacy Applications into an SOA Middleware. Satnac proceedings held in East London. South Africa.

53. Margaret, R. (2007).Requirements analysis (requirements engineering). Available from: http://searchsoftwarequality.techtarget.com/definition/requirements-analysis. (Accessed on 15-August-2012)

54. Margaret, R. (2009),systems development life cycle (SDLC). Available from: http://searchsoftwarequality.techtarget.com/definition/systems-development-life-cycle. (Accessed on 15-August.2012).

55. Mark, K. (2009). Lessons from the Past that Assist the Projects of Today to Shape the World of Tomorrow; Functional versus Non-Functional Requirements and Testing. Available from: http://lessons-from-history.com/node/83. (Accessed on 01.September 2012).

56. Mark, G. (2009). A practical Guide to Linux Commands, Editors and Shell Programming. 2Nd edition. Safari. United Kingdom.

57. Maven. (2012). Guide to Creating Archetypes. Available from: http://maven.apache.org/guides/mini/guide-creating-archetypes.html. (Accessed on 12-September-2012).

58. Michael, M. (2007). Software Engineering of Stand-alone Programs. University of Colorado, Boulder.

59. Michele E. Davis and Jon A, Phillips. (2007). Learning PhP and MySQL. A step-by step guiding to create dynamic database-driven on web site.2second edition. Published by O’Reilly Media, United States of America.

60. Muller, G. (2010). The System Architecture Process. Buskerud University College, The Netherlands. [Online]: http://www.gaudisite.nl/SystemArchitectureProcessPaper.pdf/. (Accessed on 11 June 2012).

61. Negus, C. (2007). Linux 2007 Edition. Indianapolis: Wiley Publishing, Inc.

62. Ngwenya, S. (2010).Developing a Context-Sensitive Revenue Management System for ICT4D projects in Rural Marginalized Communities. University of Fort Hare. South Africa.

63. Nichol, S. (2004). Programming with NuSOAP Using WSDL. Available from: http://www.scottnichol.com/nusoapprogwsdl.htm (Accessed 29 April 2010).

64. Nikolaj, C. (2008).On some drawbacks of the PHP platform, Proceeding CompSysTech '08 Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing Article No. 12 ACM. New York, NY, United State of America.

65. Njeje, S.G., Muyingi, H. N. and Terzoli, A. (2008). Implementing a robust, cost effective, eCommerce platform for a disadvantaged community of the Eastern Cape, South Africa: Masters Dissertation. University of Fort Hare.

66. Oracle. (2008). Ensuring Web Service Quality for Service-Oriented Architectures. Available from http://www.oracle.com/technetwork/oem/grid-control/overview/wp-ensuring-1.pdf (Accessed 22 July 2010).

67. Oracle and IDG global solutions.(2009) .PREPARING FOR MISSION-CRITICAL SOA WITH NEXT-GENERATION GOVERNANCE TECHNOLOGIES. Available from: http://www.oracle.com/us/technologies/soa/soa-governance. (Accessed on 12-March-2011).

68. Oracle and IDG global solutions. (2009). Preparing for Mission-critical SOA with Next Governance Technologies. Available from: http://www.oracle.com/us/technologies/soa/soa-governance. (Accessed on 12-March-2011).

69. Oracle Modernization. (2010) .An Executive Guide to Oracle Modernization ,Enabling Strategic Business Transformation. Available from: www.oracle.com/us/036341.pdf. (Accessed on 01-September-2012).

70. Pade, C.; Palmer, R.; Kavhai, M. & Gumbo, S.(2009). Siyakhula Living Lab: Mpume Baseline Study report. Available from: http://www.dwesa.org/sites/dwesa.org/files/baseline_report.pdf. (Accessed on 20 Jun 2010).

71. Rafael, T. (2007). Oracle Application Integration architecture. 1St edition. Oracle inc. North America.
72. Reitman, L., Ward, J. & Wilber, J. (2007). Service Oriented Architecture (SOA) and Specialized Messaging Patterns. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5751&rep=rep1&type=pdf.(Accessed 10 April 2010).
73. Richards, R. (2006). Pro PHP, XML and Web Services. California: Apress.
74. Sayo, P. (2004). Globalization and WTO: ICT, Trade and Competitiveness. In ICT Policies and e-Strategies in the Asia-Pacific. New Delhi: Elsevier. Available from: http://www.apdip.net/publications/ict4d/e-strategies.pdf (Accessed 23 July 2010).
75. Schneider, G.P. (2003). Electronic Commerce: 3rd Edition, Ph.D., CPA University of San Diego, Science Research Council & Institute of Social and Economic Research.
76. Scott, M.S. (2010). Investigation and Development of an e-Judiciary Service for a Citizen-Oriented Judiciary System for Rural Communities. Masters Dissertation. University of Fort Hare: South Africa.
77. Seagull Rocket. (2010). The Last Mile Integration: A Simpler Way to Connect Existing System to SOA.
78. Seagull, R. (2011) The Last mile of integration; A Simpler Way to Connect existing Systems to SOA, RocketSegull inc. United State of America.
79. Sklarand, D. & Adam, T. (2003). What is php? Tech Republic, A ZDNET tech community. Available from: http://articles.techrepublic.com.com/5100-22_11-5074693.html (Accessed on 10 Aug 2010.
80. Steve, E. (2007). Non-functional requirement. Department of computer science. University of Toronto.
81. Software, AG. (2008). Web Methods Application Modernization – Retail. Available from: www.softwareag.com/.../FS_App_Mod_Retail.pdf. (Accessed on 07-September-2012.)
82. Sparx Systems. (2007). UML Business process Model-Tutorials. Available from: www.sparxsystems.com.au/. (Accessed on 4 Jul 2010)

83. Spool, M. J. (2007). 7 Critical Considerations for Designing Effective Applications, Part II. Available from: http://www.uie.com/articles/designing_effective_apps_part2/ (Accessed 20-August-2012).
84. Summa. (2006). Strategy for Application Modernization, A Summa White Paper. Available from: www.summa-tech.com/.../SummaStrategyforApplication.pdf. (Accessed on 17-September 2012).
85. Tarwireyi, P.; Muyingi, H. N. & Terzoli, A. (2007). Design and Implementation of a Network Revenue Management Architecture for Marginalized Communities. Masters Dissertation. South Africa: University of Fort-Hare.
86. Tim, W. (2012). Best Practices for (Enterprise) OSGi applications. Available from. http://www.developerworks.com. (Accessed on . 20-september-2012)
87. Tutorial Point. (2010).What are Web Services?. Available from: http://www.tutorialspoint.com/webservices/what_are_web_services.htm. (Accessed on, 14-September-2012)
88. Toni, E. and Von Staden, R. (2008). Dwesa Village Connection Business Modeling Feasibility Analysis. COFISA. Ungana-Africa.
89. UNDP. (2009). Conservation and Sustainable Use in the Wild Coast. Available from: www.thegef.org/.../04-21-05%20UNDP%20PRO%20DOC.pdf. (Accessed on: 06-September-2012)
90. United Nations Conference on Trade and Development – UNCTAD. (2009). National policies to promote technological learning and innovation. In The Least developed countries report 2007, Geneva: UNCTAD: 51-90.
91. Van Buskirk, R. & Moroney, BW. (2010). Extending prototyping. IBM Systems Journa, Vol 42, No. 4, p 613-623. IBM.
92. Wertlen, R. (2010). A Design of a Middleware Solution for Connected Rural Digital Access Nodes Enabling a Multitude of Applications. Masters Dissertation. University of Fort Hare: South Africa

93. World Developments Report. (2009). Information and communication technology for development, States and Markets: Telecommunication/ICT markets and trends in Africa. ITU Switzerland

Appendix A: System installation

This section explains the installation of the system components that are not included in the main body of the thesis. This installation includes the building of the client server architecture and installation of the system components

The following technologies are the technologies that are used to install the system.

- EduBuntu 10.04
- Apache web server 2
- My SQL 5.2
- PHP 5.2.9
- Eclipse Helios
- TeleWeaver 1.2.2

All the above stated technologies are the main technologies that are used for development environment and deployment environment.

Development environment using Linux box.

The system is running on an edu Ubuntu 10.04 Linux operating system. This operating system is free and easy to obtain for usage from (http://www.ubuntu.com/). The actual installation of the operating system can be done from a Disk. During the installation, options are given to select components such as apache2, PHP5 and MySQL.

PHP installation

PHP scripting language can be installed from the Linux shell by typing the following command.

Sudo apt-get install php5.

MySQL installation

This component can also be installed using the command line below.

Sudo apt-get install mysql-server-5.

When this command line runs the component allows a developer to insert the credentials for security reasons.

PhpMyAdmin installation

To install this component, one can use the command line or can use synaptic package. This system component helps a developer to administer the database. Below is the command line that can be used to install this component.

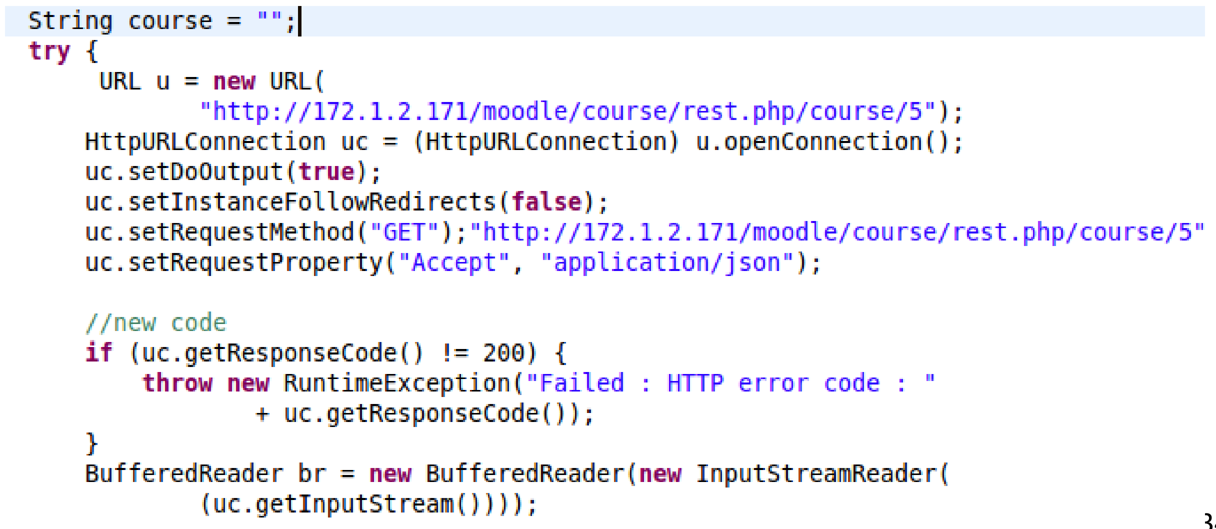
Sudo apt-get install PhpMyAdmin

During the installation of the component, the system will request for credentials, to secure my SQL database where it is necessary.

Apache installation

In this thesis we used Apache web server. The installation of this web server involves the following command.

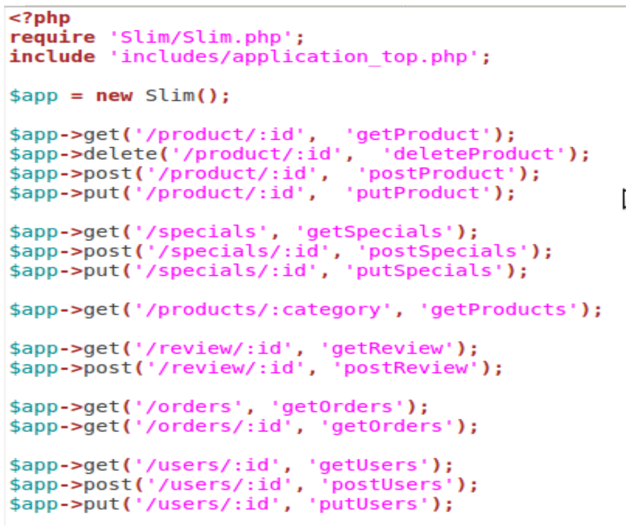
Sudo apt-get install apache 2



```
String course = "";
try {
     URL u = new URL(
            "http://172.1.2.171/moodle/course/rest.php/course/5");
    HttpURLConnection uc = (HttpURLConnection) u.openConnection();
    uc.setDoOutput(true);
    uc.setInstanceFollowRedirects(false);
    uc.setRequestMethod("GET");"http://172.1.2.171/moodle/course/rest.php/course/5"
    uc.setRequestProperty("Accept", "application/json");

    //new code
    if (uc.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : "
                + uc.getResponseCode());
    }
    BufferedReader br = new BufferedReader(new InputStreamReader(
            (uc.getInputStream())));
```

Listing 9.1 Connection between TeleWeaver Middleware and Moodle Application.

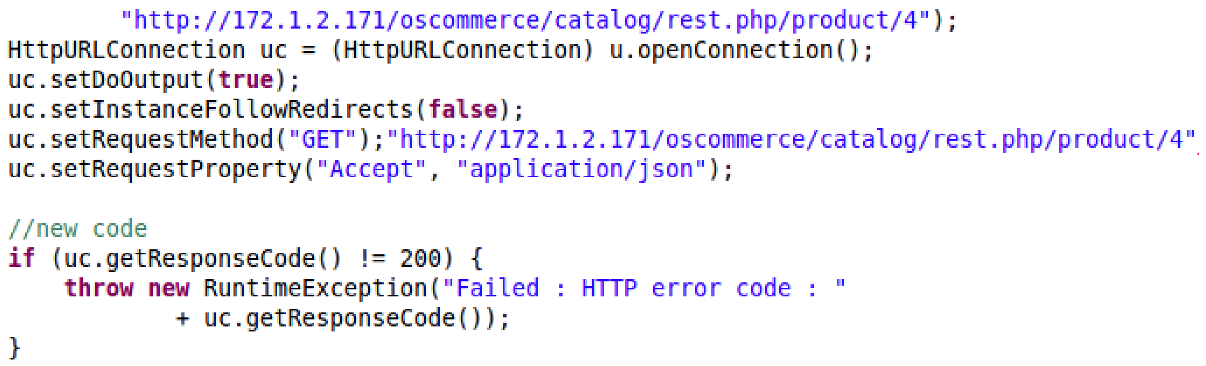
The above listing elaborates how Moodle is integrated into TeleWeaver when using slim API. The above listing shows the importance of URI when using RESTful web service. This shows major connection between the legacy applications and a TeleWeaver middleware. This is an example of the external integration of legacy services. However the method of the service has to be specified in the code. The route of the components is a crucial factor when integrating legacy application using REST service. The response type of the component needs to be specified. However if the response method and a request method are not specified on the code, the integration will not be successful. An exception has to be thrown to catch any error in the system integration.



```php
<?php
require 'Slim/Slim.php';
include 'includes/application_top.php';

$app = new Slim();

$app->get('/product/:id',  'getProduct');
$app->delete('/product/:id',  'deleteProduct');
$app->post('/product/:id',  'postProduct');
$app->put('/product/:id',  'putProduct');

$app->get('/specials', 'getSpecials');
$app->post('/specials/:id', 'postSpecials');
$app->put('/specials/:id', 'putSpecials');

$app->get('/products/:category', 'getProducts');

$app->get('/review/:id', 'getReview');
$app->post('/review/:id', 'postReview');

$app->get('/orders', 'getOrders');
$app->get('/orders/:id', 'getOrders');

$app->get('/users/:id', 'getUsers');
$app->post('/users/:id', 'postUsers');
$app->put('/users/:id', 'putUsers');
```

Listing 9.2 Slim API

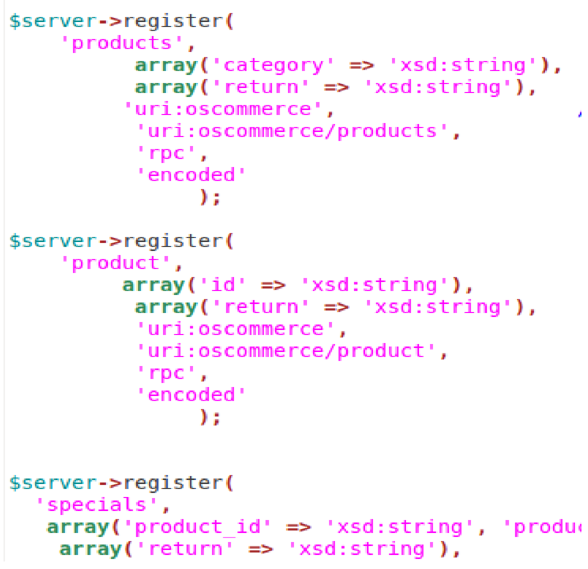
The listing 9.2 illustrate how methods are registered when using Slim API. This shows the method and the route of the components. When using the Slim API the route has to be specified, the API need to be alert of which service uses which method such as GET, POST, PUT or DELETE method.



```
        "http://172.1.2.171/oscommerce/catalog/rest.php/product/4");
HttpURLConnection uc = (HttpURLConnection) u.openConnection();
uc.setDoOutput(true);
uc.setInstanceFollowRedirects(false);
uc.setRequestMethod("GET");"http://172.1.2.171/oscommerce/catalog/rest.php/product/4"
uc.setRequestProperty("Accept", "application/json");

//new code
if (uc.getResponseCode() != 200) {
    throw new RuntimeException("Failed : HTTP error code : "
            + uc.getResponseCode());
}
```

Listing 9.3. The e-Commerce connection with TeleWeaver Middleware

This listing shows the connection between e-Commerce application and TeleWeaver middleware. This code illustrates how e-Commerce is integrated in TeleWeaver Middleware when using slim API; however the route and methods has to be specified. This is the similar code with Moodle applications, since these applications both uses PHP language, the mechanism of integrating them is similar.



```
$server->register(
    'products',
          array('category' => 'xsd:string'),
          array('return' => 'xsd:string'),
         'uri:oscommerce',
          'uri:oscommerce/products',
          'rpc',
          'encoded'
               );

$server->register(
    'product',
         array('id' => 'xsd:string'),
          array('return' => 'xsd:string'),
          'uri:oscommerce',
          'uri:oscommerce/product',
          'rpc',
          'encoded'
               );


$server->register(
  'specials',
   array('product_id' => 'xsd:string', 'produ
    array('return' => 'xsd:string'),
```

Listing 9.4 Nu-SOAP services

The Nu-SOAP is an adapter model that is used been used to integrate the two stated legacy application into SOA middleware, however this Moodle was not successful due to the complications that were encountered during the integration process. The available interfaces on TeleWeaver middleware they did not support the Nu-SOAP model for integration process. The e-Commerce and Moodle components were already registered in Nu-SOAP server and they were functioning properly, however the problem was to integrate the Moodle into TeleWeaver middleware. That resulted to development of new mechanism which is slim API, which supported the integration process, as TeleWeaver supported the API.

TeleWeaver installation

TeleWeaver is an SOA middleware that has been developed to integrate legacy applications under SLL context. As stated in Chapter Two, TeleWeaver supports several technologies and has interfaces that support external integration of legacy application. TeleWeaver is installed in Linux system.